НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено

Завідувач кафедри

_____ О.В. Коваль
(підпис) (ініціали, прізвище)

“___”_____2019р.

ДИПЛОМНА РОБОТА
на здобуття ступеня бакалавра

з напряму підготовки 6.050103 “ Програмна інженерія “

на тему “Веб-інтерфейс візуалізації геопросторових даних для мікросервісної платформи“

Виконав (-ла): студент (-ка) 4 курсу, групи ТВ-51

Жирнов Андрій Юрійович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник доцент, к.т.н, Смаковський Д.С. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший , бакалаврський

Напрям підготовки 6.050103 “ Програмна інженерія”

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2019р.

ЗАВДАННЯ
на дипломну роботу студенту

Жирнов Андрій Юрійович
(прізвище, ім’я, по батькові)

1. Тема роботи “Веб-інтерфейс візуалізації геопросторових даних для мікросервісної платформи“

керівник роботи Смаковський Денис Сергійович, доцент, к.т.н.
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”____ 201__р. № __

2. Строк подання студентом роботи _____

3. Вихідні дані до роботи Форма реалізації – веб-інтерфейс з візуалізованими геопросторовими даними на інтерактивній карті

4.Зміст розрахунково-пояснювальної записки (перелік завдань, які потрібно розробити) Проаналізувати основні проблеми відображення і візуалізації геопросторових даних для аграрного сектору, дослідити існуючі програмні засоби для роботи з геопросторовими даними та їх відображенням. Розробити веб-інтерфейс, що буде надавати можливості обробки та візуалізації геоданих.

5. Перелік ілюстративного матеріалу
«Актуальність», «Вступ», «Існуючі програмні рішення», «Постановка задачі», «Взаємодія веб-інтерфейсу з елементами мікросервісної платформи», «Взаємодія користувача з системою», «Основні засоби розробки», «Створення адаптивного веб-інтерфейсу», «Інтерфейс системи», «Висновки»

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата	
		завдання видав	Завдання прийняв

7. Дата видачі завдання ”__”_____201__р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Вивчення та аналіз задачі	01.12.2018-31.03.2019	
2	Розробка архітектури та загальної структури системи	01-18.04.2019	
3.	Розробка структур окремих підсистем	19-25.04.2019	
4.	Програмна реалізація системи	26.04-13.05.2019	
5.	Оформлення пояснювальної записки	06.05-01.06.2019	
6.	Захист програмного продукту	25.05.2019	
7.	Передзахист	01.06.2019	
8.	Захист	17.06.2019	

Студент _____ Жирнов А.Ю.
(підпис) (прізвище та ініціали,)

Керівник роботи _____ Смаковський Д.С.
(підпис) (прізвище та ініціали,)

АНОТАЦІЯ

Метою дипломної роботи є створення зручного web-інтерфейсу для візуалізації геопросторових даних в аграрному секторі.

Об’єктом дослідження є геопросторові дані та методи їх відображення. Було виконано огляд існуючих програмних застосунків для візуалізації геопросторових даних, виявлено їх переваги та недоліки. Створено web-інтерфейс, що вирішує проблеми обробки та відображення геоданих для сільськогосподарського сектору.

Створена програмна система може бути використана у роботі підприємців-аграріїв для автоматизації ведення стану полів.

Загальний обсяг роботи: 50 сторінка, 20 ілюстрацій, 16 бібліографічних посилань та 3 додатки.

Ключові слова: геопросторові дані, веб-інтерфейс, аграрний сектор, візуалізація даних, мікросервісна платформа, інтерактивна карта.

ABSTRACT

The purpose of the thesis is to create a convenient web-interface for visualizing geospatial data in the agrarian sector.

The object of the study is geospatial data and methods for their reflection. A review of existing software applications for visualization of geospatial data was performed, their advantages and disadvantages were identified. A web-based interface has been created that solves the problem of processing and displaying location data for the agricultural sector.

The created software system can be used in the work of entrepreneurs-agrarians to automate the state of the fields.

Total volume of the paper: 50 pages, 20 illustrations, bibliography links and 3 appendixes.

Key words: geospatial data, web interface, agrarian sector, data visualization, micro-service platform, interactive map.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

1. СУБД — система управління базами даних; комплект даних та програмних інструментів для доступу до них.
2. ГІС — геоінформаційна система; поєднує карту території та різні дані, в основному табличного типу.
3. GSM-модуль — плата, що дозволяє виходити в інтернет за допомогою мережі GPRS.
4. NDVI — індекс, за яким можна судити про розвиток зеленої маси рослин під час вегетації.
5. AJAX — Asynchronous JavaScript And XML (асинхроний JavaScript та XML.

ЗМІСТ

ВСТУП

Україна — країна, що має величезні перспективи для розвитку аграрного сектору. Широкі лани, родючі чорноземи, помірний клімат та велика кількість спеціалістів — дають можливість створити сильну галузь, яка й на сьогодні є однією з провідних в нашій державі. За статистикою минулого року, саме сільськогосподарські угіддя, на яких працевлаштовано більше трьох мільйонів українців, принесли 40% всієї валютної виручки нашої країни.

Важливим аспектом родючості грунту є рівень щільності грунту в різних шарах землі. Поживні речовини гірше доходять до рослин при переущільненні грунту й корінню складніше розростатись в такій землі, тож кожен шануючий себе фермер слідкує за своєю землею.

Для дослідження цієї характеристики використовується пенетрометр, що виконує точкову перевірку грунту. Він виконує 19 вимірів, з кроком 2.5 сантиметра в глибину, в радіусі близько 1.5 метра. Такий прилад існує давно, й ним активно користуються.

Постає проблема для контролю якості грунту на всій площі поля, збіру даних в одному місці для їх аналізу. На сьогодні немає публічної системи, що могла б допомогти аграріям в цьому.

Більшість великих аграрних підприємств має свої технології збору та обробки даних. Деякі тримають свій відділ спеціалістів, що займаються аналізом та прогнозуваням стану та якості землі. Та не кожна компанія може дозволити собі тримати штатних співробітників або замовити власну систему аналізу та збору даних, осоливо дрібні фермери.

Закономірно виникає необхідність створення зручного рішення, що дозволятиме кожному підприємцю аграрної галузі забезпечити собі вільний доступ до всіх своїх показників у зручному форматі.

Для розробки такого сервісу необхідна команда програмістів, серед яких буде людина, що має досвід у роботі з ГІС системами та даними, часто пошук таких спеціалістів викликає складнощі. Тож для створення такого сервісу необхідно

зібрати та проаналізувати існуючі системи, зібрати дані та алгоритми для роботи з геоінформаційними даними для їх подальшого відображення в особистому кабінеті користувача на інтерактивній карті.

Пропонується розробка мікросервісної системи, що матиме змогу відправляти дані з пенетрометра одразу на сервер, для їх подальшого аналізу, пакування в зручні звіти, що будуть наочно відображати всі виміри по їх координатам, полям та часових проміжках.

Веб-інтерфейс продукту повинен дати змогу користувачу в зручному форматі отримувати виміри по своїм полям, границі яких теж можна задати прямо у веб-інтерфейсі додатку.

Для рішення цього завдання була зібрана команда молодих спеціалістів, кожному з яких доручено створення окремого мікросервісу для їх кінцевого поєднання в одну велику платформу, що буде відповідати всім поставленим вимогам.

1. ЗАДАЧА РОЗРОБКИ ВЕБ-ІНТЕРФЕЙСУ ВІЗУАЛІЗАЦІЇ ГЕОПРОСТОРОВИХ ДАНИХ ДЛЯ МІКРОСЕРВІСНОЇ ПЛАТФОРМИ

На аграрному ринку України значний відсоток великих компаній, як українських, так і іноземних, що мають у власності маштабні території у володіні. Такі холдинги володіють великою кількістю сучасної аграрної техніки, що позбавила людей необхідності обробляти поля вручну. Та все таки сільсько-господарський бізнес досить консервативна галузь, особливо в сфері впроваження інформаційних технологій в свої господарства.

На сьогоднішній день IT-компанії України теж не сильно зацікавлені в цій сфері, або ж працюють цілenаправленно на якийсь великий аграрний холдинг, створюючи продукти безпосередньо під клієнта. Рядовий фермер або дрібний підприємець не має достатніх знань та коштів для замовлення програмного продукту під свої потребності. Ті ж, хто має такі ресурси може навіть не задумуватись над модернізацієї свого господарства, та автоматизації рутинних процесів.

Хоч і статистика каже про те, що на сьогоднішній день доволі таки великий відсоток сучасних українських стартапів націлений саме на проекти в аграрній області, наприклад, на конкурсі стартапів Sikorsky Challenge (рисунок 1.1)[1] було представлено багато цікавих рішень для сільсько-господського сектору, та більшість з них залишилась на стадії прототипу.

Можливо більшість з них не знайшла достатнього ентузіазму та інвестицій зі сторони, щоб довести ідеї до реального продукту, та це лише підтверджує, що аграрії України не сильно зацікавлені в самостійному впровадженні інформаційних технологій в свої підприємства, або ж вважають це дорогим та не настільки необхідним у виробництві. При цьому вони не розуміють, наскільки будь-яка автоматизація виробництва полегшує роботу, систематизує дані та спрощує роботу з

підприємством загалом, позбавляючи від необхідності виконання циклічних, рутинних операцій.

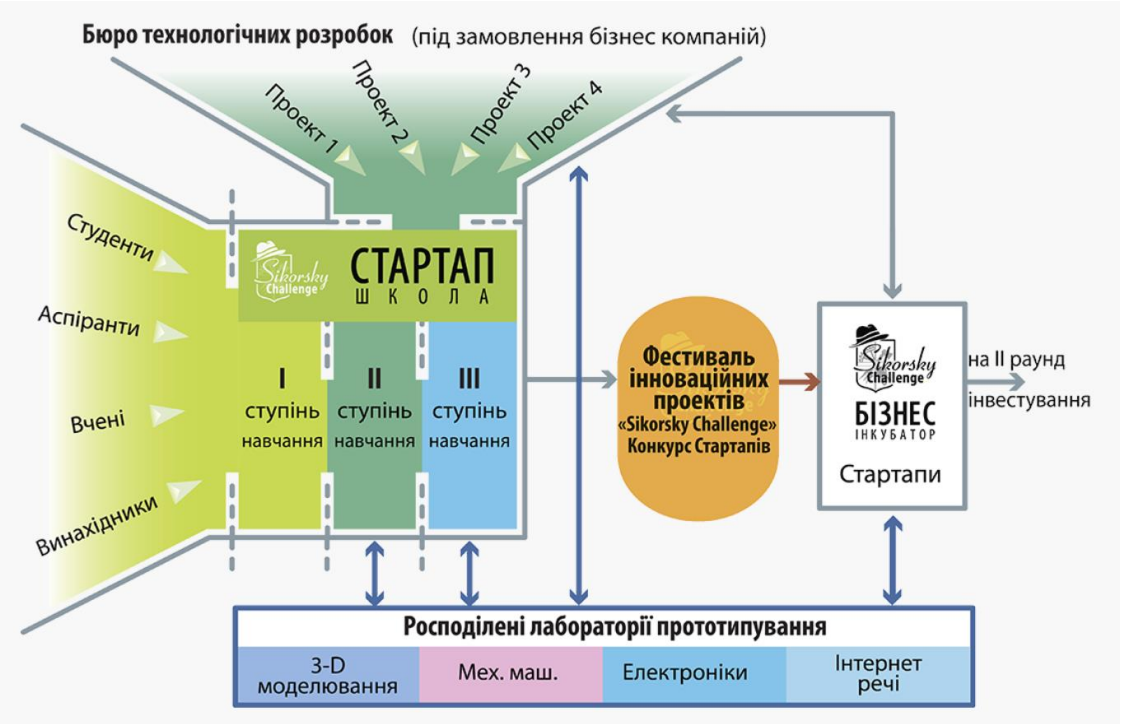

Рисунок 1.1 — Алгоритм роботи школи стартапів Sikorsky Challenge

Проаналізувавши дані, що приведені вище, було поставлено ряд вимог до кінцевого продукту, що представлятиме собою: просту у розумінні, легку у використанні та зручну для впровадження в будь-яке господарство систему.

Вона повинна надавати сховище для всіх вимірів та геоінформаційних даних, будувати звіти, аналізувати рівень щільності грунту. Поєднувати весь функціонал має веб-інтерфейс із адаптивною верстkою.

1.1 Створення зрозумілого інтерфейсу для підприємця-фермера

Будь-який власник землі під аграрний бізнес знає — щільність грунту є одним з основним показників якості землі, який варто перевірити перед засадженням сільсько-господарськими культурами. Зайве ущільнення грунту обмежує нормальне проникнення води, повітря та поживних речовин до кореневої системи, що

призводить до послаблення ростових процесів всієї рослини і зниження врожайності. У таких ґрунтах знижується швидкість повітрообміну і мінералізації азоту.

Основна проблема полягає в тому, що дані грунтового пенетрометру відображаються на екрані, безпосередньо в місці виміру, яке має радіус близько метру. Якось зберігати чи переносити дані на комп'ютер створює низку проблем, не кажучи про їх зручну структуризацію та відображення. Тобто підприємець, у власності якого великі ділянки землі, вимушений або самостійно та вручну збирати дані, по різним частинам полів, якимось чином це структуруючи, звісно ж, без достатньої точності. В подальшому для складення повної картини стану грунту на своїх територіях йому необхідно перенести дані на карту, визначити проблемні місця та лише потім планувати міри для покращення грунтів, в разі виявлення якихось проблем.

Для автоматизація та спрощення рутинного процесу й було призвано створити веб-сайт для відображення даних по вимірам безпосередньо на карті, з можливістю мануального задання координат області для відображення даних в ній.

Веб-інтерфейс повинен відповідати наступим вимогам:

1. Інтерактивна карта Землі, з можливістю динамічного редагування та оновлення.
2. Можливість задання нового поля безпесередньо на карті, просто виділяючи необхідну область.
3. Створення нового поля з масиву координат — вершин кінцевого полігону.
4. Візуалізація даних вимірів з бази даних по їх координатам на карті користувача.
5. Можливість скачати звіт пенетрометру.
6. Інструмент для редагування поля безпосередньо на інтерактивній карті веб-застосунку.
7. Кольорове заповнення поля або точки виміру, в залежності від середнього значення вимірів в даному місці.
8. Зручний та зрозумілий для будь-кого, навіть початківця, користувацький

інтерфейс.

9. Адаптивне відображення для будь-якого пристрою.

Кінцева система складається з низки мікросервісів, кожен з яких має свій функціонал та відповідає за окрему групу процесів для аналізу та представлення вхідних даних. Весь сервіс поєднаний між собою за допомогою зовнішніх програмних інтерфейсів та базою даних, в якій зберігаються дані вимірів для кожного клієнту та межі полів, заданих користувачем через веб-інтерфейс.

1.2 Створення гнучкої системної архітектури для подальшої модифікації сервісу.

Для вимірів щільності грунту використовується пристрій — пенетрометр, що додатково оснащено GSM-модулем, що забезпечує передачу даних вимірів та геолокації безпосередньо на сервер.

Сам пенетрометр може мати два типи накінечників: “1/2” та “3/4” для різних типів грунтів. Він виконує 19 замірів в глибину з кроком в 2.5 сантиметра і повертає значення щільності грунту по кожному шару.

Отже має бути створений клас приладу, який матиме властивості, що свідчитимуть про тип накінечника, номер приладу, компанію виробника та кастомний опис від користувача.

Зручно було б розділити обробку та аналіз різних даних у різні модулі системи, наприклад — окремі мікросервіси. Та окрім зручності мікросервісна будова дає переваги й в інших сценаріях.

Система не повинна бути обмеженою та легко піддаватись модифікації або розширенню. Щоб в подальшому, при виникнені необхідності додання того чи іншого функціоналу це можна було зробити без проблем, інтегрувавши новий модуль, не вносячи зміни до існуючого коду серверу, лише додавши нові елементи інтерфейсу для сайту.

1.3 Проектування сервісів обробки даних з пристрою для їх подальшого відображення

Сервіс по створенню звітів повинен систематизувати дані, створюючи звіти у форматах: .xlsx, .xml та .pdf, також у звітах мають знаходиться представлення даних у вигляді графіків. Звіт передаватиметься до веб-інтерфейсу за допомогою HTTP-запиту. Користувач зможе скачати його за вибором у форматах: .xlsx, .xml або .pdf.

Сервіс полів та вимірів виконуватиме задачу отримування інформації про зроблені виміри, по даті та місцю, та готові поля, що задані користувачем раніше. Усі ці відомості підтягуватимуться з бази даних системи, в якій зберігатиметься вся інформація користувача. Найважливіша частина системи, що є його ядром, яке відповідає за основні дані, тому повина бути розроблена з врахуванням всіх можливих помилок та складнощів.

Отже загальна структура сервісу (рисунок 1.2) повина складатись з декількох мікросервісів, що будуть побудовані на мові Java та веб-інтерфейсу для зручного відображення даних кінцевому користувачеві.

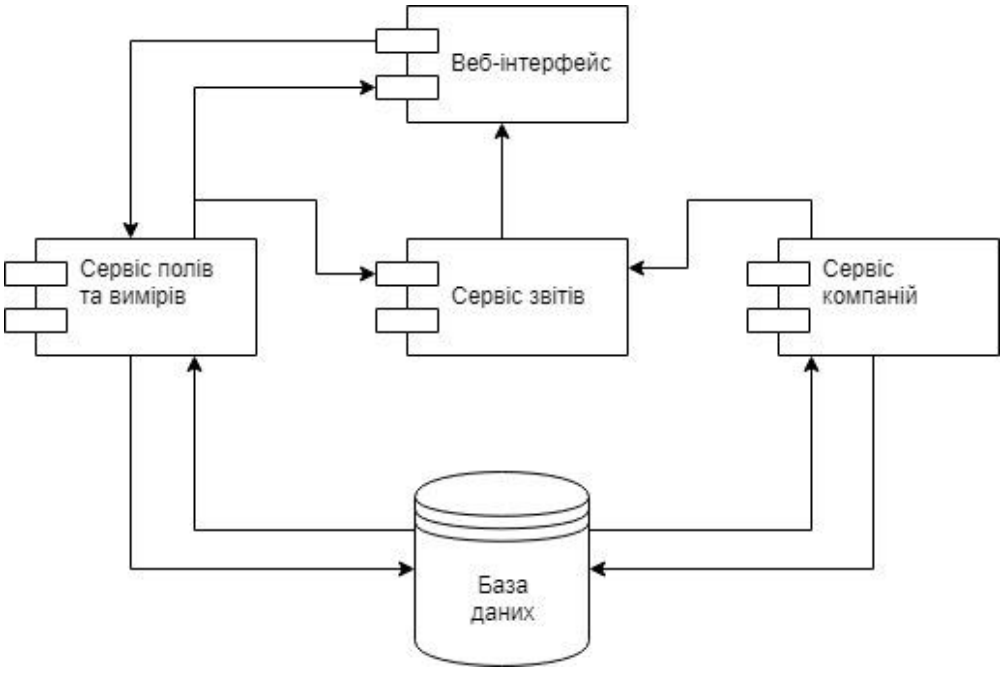

Рисунок 1.2 — Схема взаємодії частин порталу

Для керуванням даними з бази даних пропонується використати СУБД

PostgreSQL, що зарекомендувала себе в багатьох великих проектах, показавши свою стабільність та дієздатність.

Для зв’зку веб-відображення системи та серверної частини зазвичай використовується зв’язок за допомогою протоколу передачі даних HTTP[3]. Це стара та провірена часом технологія, що використовується у кожному веб-сайті чи веб-застосунку.

Перед беспосереднім відображенням сайту на екрані користувача — браузер відправляє свій запит до серверної частини, щоб отримати дані для відображення. Разом із зручністю способу, постає інше питання — швидкість відображення, адже вона напряму впливає на досвід, що отримає людина після користування даною веб-системою. Розуміючи архітектуру взаємодії двох частин клієнт-серверного застосунку, можна й робити кроки, щоб пришвидшити роботу веб-інтерфейсу з легкістю.

Браузер називають HTTP клієнтом, тому що він надсилає запит на отримання даних за допомогою HTTP протоколу, а сервер в свою чергу повертає необхідні дані назад на сервер.

Загалом, HTTP — це протокол передачі гіпертексту, що розроблений дуже давно як основа для всієї World Wide Web і використовується по наш час для передачі будь-яких даних, таких як HTML, CSS, Javascript та навіть звичайних картинок[14].

Та основна проблема такого протоколу — його синхроність, тобто зв’язок з сервером встановлюється чередою HTTP-запитів і відповідей, тобто в цей момент обміну, користувач вимушений просто чекати і сподіватись, що пропускна здатність його інтернет-з’єднання не змусить його очікувати занадто довго, щоб це відбило бажання до подальшої роботи з сайтом.

В нашому ж випадку ми змушені створити систему, що буде постійно обмінюватись даними з сервером. При цьому, не змушуючи користувача очікувати довгої відповіді від сервера. На карті інтерфейсу, при завантаженні, мають відображатись місця усіх вимірів й відповідно межі полів, що будуть знаходитись у базі даних сервісу. Ця інформація отримуватиметься за допомогою HTTP-запитів до

сервісу, що відповідає за поля та виміри, звідки надходитеме відповідь у форматі JSON, який в свою чергу оброблюватиметься засобами мови Javascript, для подальшого відображення на веб-сторінці сервісу. Також всі ці дані повині мати здатність до редагування, та динамічної синхронізації з серверною частиною системи.

Для вирішення такої проблеми було створено технологію AJAX (Рисунок 1.3), що чудово справляється з асинхроністю.

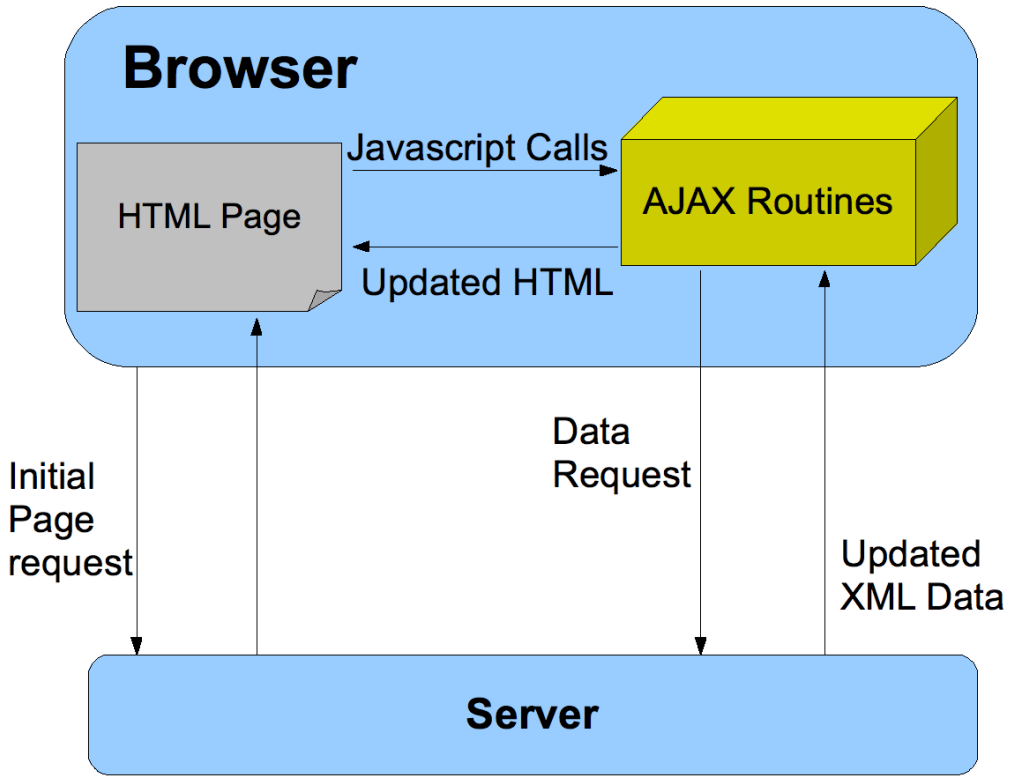

Рисунок 1.3 — Схема взаємодії HTML-сторінки з сервером засобами AJAX

AJAX — це мрія будь-якого розробника, тому що ця технологія надає наступні можливості:

1. Читати дані з серверу, навіть після того, як цільова веб-сторінка вже була завантажена.
2. Модифікувати та змінювати дані самої веб-сторінки, без її безпосереднього перезавантаження.
3. Відправляти дані на сервер в фоновому режимі.

Ця технологія побудована на стандартному класі браузера — XMLHttpRequest, що дозволяє працювати з HTTP-запитами. Разом з ним весь потенціал технології розкривають можливості JavaScript та його інструменти для керування всім DOM-деревом сайту.

Асинхронні запити AJAX дають можливість сайтам динамічно оновлюватись, обмінюючись інформацією з сервером, при цьому не змушуючи користувача чекати перезавантаження сторінки чи завантаження даних. Така технологія повністю вирішує проблему динамічного відображення геопросторових вимірів на інтерактивній карті.

1.4 Висновки до розділу

Була поставлена задача і проаналізовано можливі проблеми при створенні продукту. Для подальшого проектування і розробки веб-додатку виділено основні цілі та умови, що мають бути виконані для того, щоб кінцевий користувач отримав всі необхідні функції

2. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ АВТОМАТИЗАЦІЇ АГРАРНИХ ПРОЦЕСІВ ТА ВІЗУАЛІЗАЦІЇ ГЕОПРОСТОРОВИХ ДАНИХ НА КАРТІ

Люди з покон віків почали займатись обробкою землі та вирощенням різного роду сільськогосподарських культур для створення постійного та стабільного ресурсу отримання їжі. Аграрний сектор по сьогоднішній день є одним з основних джерел харчування людства, так що важко переоцінити його важливість для кожного з нас.

З кожним етапом розвитку нашої цивілізації, знаряддя праці покращувались, модернізувались, методи обробки землі змінювались. Кожен новий винахід чи підхід робив міні-революцію в аграрній промисловості. А за останні роки, коли технологічний процес дуже стрімкий важко навіть вслідкувати за усіма новими трендами в галузі.

Поєднання ГІС-технологій з агрономією дає дуже плідний результат, адже це допомагає краще слідувати за своїми територіями, бачити і аналізувати необхідну інформацію про стан і продуктивність полів. Знаходити проблеми з врожаями значно простіше, маючи дані, що лаконічно візуалізовані на карті, що знаходиться прямо на екрані комп’ютера.

2.1 Геоінформаційні системи та агрономія

Велика кількість задіяних у сільськогосподарській діяльності полів, людей, техніки та інших ресурсів приводить людство до знаходження якісно нових засобів керування земельними ресурсами України. В еру інформаційних технологій, вони як ніщо інше зможуть кардинально змінити консервативні правила ведення аграрного бізнесу.

Геоінформаційні системи — це сучасні комп’ютерні системи, що дозволяють

поєднувати обробку, аналіз та структуризацію даних з їх кінцевим відображенням на об’єкті візуалізації. Кожна така система призначена для фіксування, редагування та аналізування певних географічних даних. Тобто для роботи з такими проектами, розробник повинен мати хоча би базові пізнання у географічних науках, щоб вірно спроектувати всі елементи проекту[15].

Одним з найбільш перспективних напрямків сільськогосподарського виробництва є використання інформаційних систем на основі геоінформаційних технологій. За допомогою таких систем можна вирішити такі завдання:

1. Інформація про підтримку прийняття рішень;
2. Планування господарств;
3. Моніторинг господарств та умов врожаю;
4. Прогноз прибутковості та оцінка втрат;
5. Планування, моніторинг та аналіз використання технологій.
6. Дивіться кожну з них більш уважно.
7. Інформаційна підтримка прийняття рішень

Для того, щоб надати менеджерам достатньо інформації для прийняття управлінських рішень, необхідно створити базу даних, що міститиме дані:

1. Цифрову модель місцевості, де здійснюється сільськогосподарська діяльність.
2. Інформація про стан грунту, його щільність, однорідність чи неоднорідність.
3. Інформація про поля, особливості їх георозташування та тип землі в цьому місці.
4. Картки врожаїв за рік.
5. Історія обробки на полях тощо.

Для вирішення задач комплексного аналізу в сільському господарстві використовуються електронні карти з результатами супутникових геодезичних вимірювань. Використання таких методів дозволяє отримати детальну інформацію про великі території.

Практика показує, що термін окупності інвестицій, орієнтованих на реалізацію

інтеграції ГІС, становить від 1 до 3 років, залежно від розміру розгорнутої системи, і початковий ефект від впровадження системи можна ясно побачити в кінці першого сезону після впровадження. Конкурентоспроможність зростає з прибутковістю бізнесу через зниження витрат і більш ефективне використання наявних ресурсів.

Таким чином, створення системи інформаційної підтримки процесів прийняття рішень на основі ГІС-технологій дозволяє підвищити загальну ефективність сільськогосподарського виробництва за рахунок надання актуальної аналітичної інформації по всьому комплексу необхідних параметрів для прийняття оптимальних і своєчасних управлінських рішень.

2.2 Переущільнений грунт

Грунт має різні фізико-механічні властивості, одна з яких - щільність. Визначення щільності грунту вважається однією з основних фізичних характеристик грунту, що описує його стан. Для визначення щільності ґрунту в лабораторних умовах необхідно розрахувати відношення маси речовини до її обсягу. Проводячи інженерно-геологічні дослідження, необхідно розрахувати деякі показники, що впливають на кінцевий результат при визначенні щільності зразка: щільність грунту, щільність твердих частинок грунту і щільність скелета грунту.

Як зазначалося вище, для того, щоб максимально точно визначити щільність ґрунту, необхідно розрахувати інші показники. Тверді частинки в грунті є мінералами, органічними і органо-мінеральними речовинами. Також можна спочатку визначити щільність вологого ґрунту (з природною вологістю). Природно, що при збільшенні вмісту води в ґрунті щільність ґрунту також зростає. Під скелетом грунту розуміється його сухий стан. Знаючи всі ці показники, фахівці можуть визначити загальну щільність грунту[16].

2.2.1 Фактори впливу на щільність грунту

Сьогодні, з розвитком нанотехнологій, складні математичні розрахунки більше не потрібні. Визначення щільності ґрунту стало можливим завдяки

спеціальному сучасному обладнанню. Люди неодноразово стикалася з проведенням лабораторних досліджень ґрунту для проектування доріг, фундаментів будинків і споруд, укладання комунікацій тощо. Важливо зрозуміти, чому така робота виконується і мати достатній досвід у цій галузі. Практично в будь-якій конструкції ущільнення ґрунту є одним з основних показників, які згодом впливають на продуктивність інженерних споруд. Несуча здатність грунтів може різко змінити рішення дизайнера щодо вибору фундаменту для будівлі або споруди, а також матеріалу.

З'єднання грунту (адгезія) прямо пропорційно його щільності. При виявленні слабких ґрунтів, схильних до деформації, переміщення і набухання на місці, архітектор повинен серйозно задуматися про свої подальші дії. Найменша помилка може викликати багато проблем.

Від щільності грунту залежить її проникність. І, як відомо, наявність вологи в ґрунті, а також її чудові вологопоглинаючі властивості природних вод (наприклад, талі) можуть призвести до усадки грунту, а взимку до морозного обмерзання. Знаючи ці показники, можна, не чекаючи таких неприємностей, як затоплення або руйнування споруд, попередньо розрахувати водостічну систему для вашого будинку (будівлі) і тоді ніяке затоплення не страшно[11].

В аграрному секторі ж переущільнення грунтів призводить до гірших врожаїх, тому що рослини не отримують необхідної вологи та поживних речовин, що не можуть потрапити до коріння. В пухкому ж грунті рослина навпаки погано тримається за рахунок того, що кореневище не може надійно зафіксуватись в грунтовій породі.

Тож щільність грунту важлива не лише в сільсько-господарській сфері, а й в будівництві, прокладенні зв’язку тощо.

2.3 Аналіз поширення інформаційних технологій у сфері аграрного бізнесу

По мірі аналізу ринку хмарних веб-застосунів було виявлено, що подібні

системи зовсім не поширенні, тож прямих аналогів та конкурентів віднайти не вдалось.

В основному це пов’язано з тим, що більшість таких систем розроблюються приватно під кожного клієнта, в ролі якого виступають великі агрохолдинги, сільськогосподарські угіддя різних масштабів, що мають бюджет для замовлення таких систем.

Можна знайти інформацію про велику кількість аграрних IT-стартапів, які мали схожі ідеї по автоматизації сільськогосподарської сфери, та всі ці проекти було заморожено ще до стадії виходу першого прототипу.

Можливо також, що така ситуація пояснюється низьким рівнем інформатизації в усій сфері, та неготовністю рядових фермерів впроваджувати важкі та незрозумілі системи у своє, давно вже налагоджене, господарство. Та скоріш за все — це наслідок відсутності зручного та простого у використанні програмного забезпечення, що справді принесе позитивні зміни для їх бізнесу при цьому не б’ючи по матеріальному становищу клієнта.

2.4 Огляд готових рішень для візуалізації геопросторових даних у веб-середовищі.

Хоч у профільній сфері і немає публічних аналогів, та застосунків для візуалізації даних, але такі засоби поширені у інших галузях, що потребують відображення певних вимірів на карті.

— Agro API — реалізація програмного інтерфейсу для візуалізації температурних даних по геолокації. Веб-застосунок пропонує вузький набір можливостей, схожих по типу з розроблюємою системою. Має велику базу температурних та фотографічних даних планети, які можна використовувати для підключення в свої проекти та застосунки (рисунок 2.1). Розповсюджується по моделі місячної підписки на сервіс з декількома варіантами доступу.

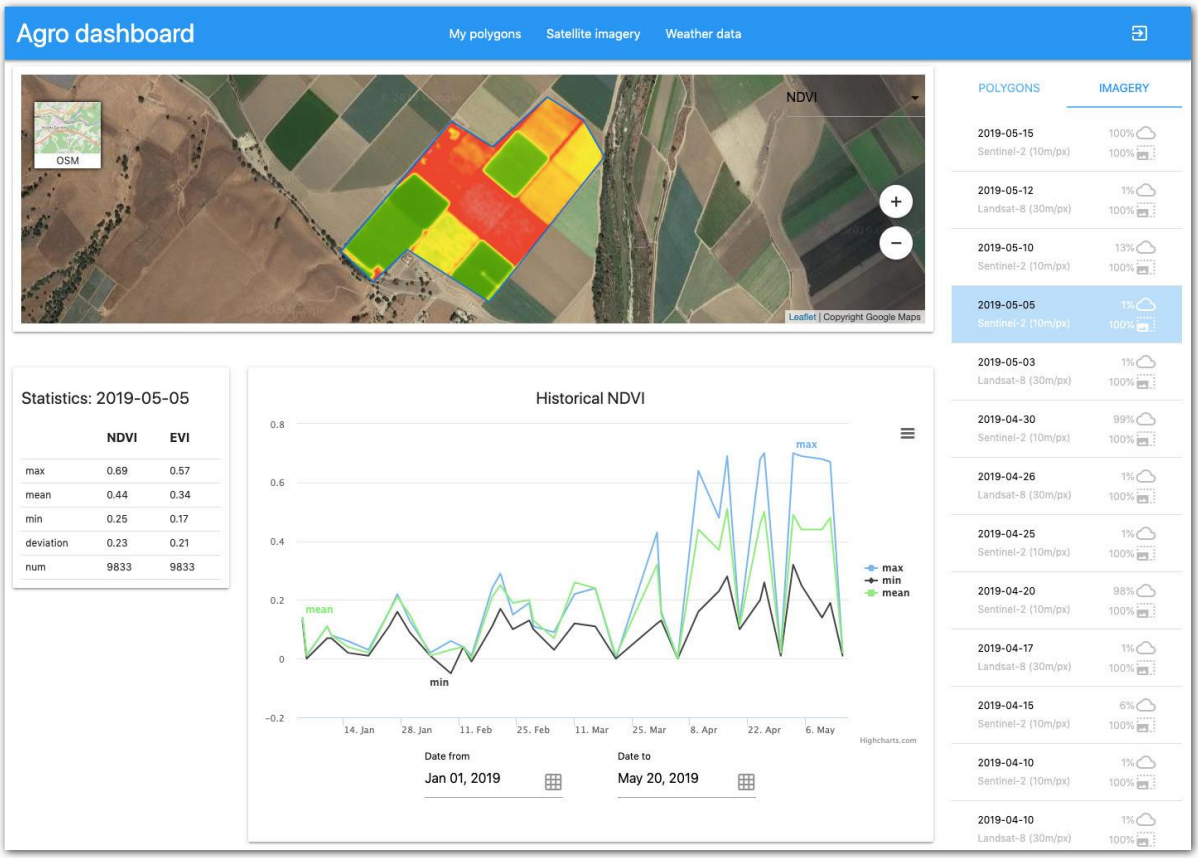

Рисунок 2.1 — Особистий кабінет Agro API

Ця система надає такий список можливостей[2]:

1. Доступ до нових супутникових знімків кожні 2-4 дні.
2. Індекси рослинності NDVI та EVI та її статистика.
3. Погодинний і щоденний прогноз погоди.
4. Поточні та історичні дані погоди.
5. Накопичуються температурні показники і опади.
6. Історична діаграма NDVI.

Загалом продукт надає доступ до трьох видів API:

1. API погодних даних.
2. API супутникових знімків.
3. API для машиного навчання.

Кожен продукт самостійний і його можна використовувати окремо від інших без проблем.

— ArcGIS Online — продукт, що входить у велике сімейство

геоінформаційних продуктів від американської компанії ESRI. ArcGIS Online надає місце для розміщення в мережі ваших карт з відповідною графічною інформацією. Також можливість обміну нею з користувачами. Це система управління географічною інформацією, що дозволяє обмінюватися вашим вмістом, а також розміщувати його в ГІС додатках і на веб-сайтах кінцевих користувачів (рисунок 2.2). Дуже гнучкий додаток, що може допомогти у вирішенні майже будь-якої проблеми, пов’язаної з візуалізацією даних на карті.

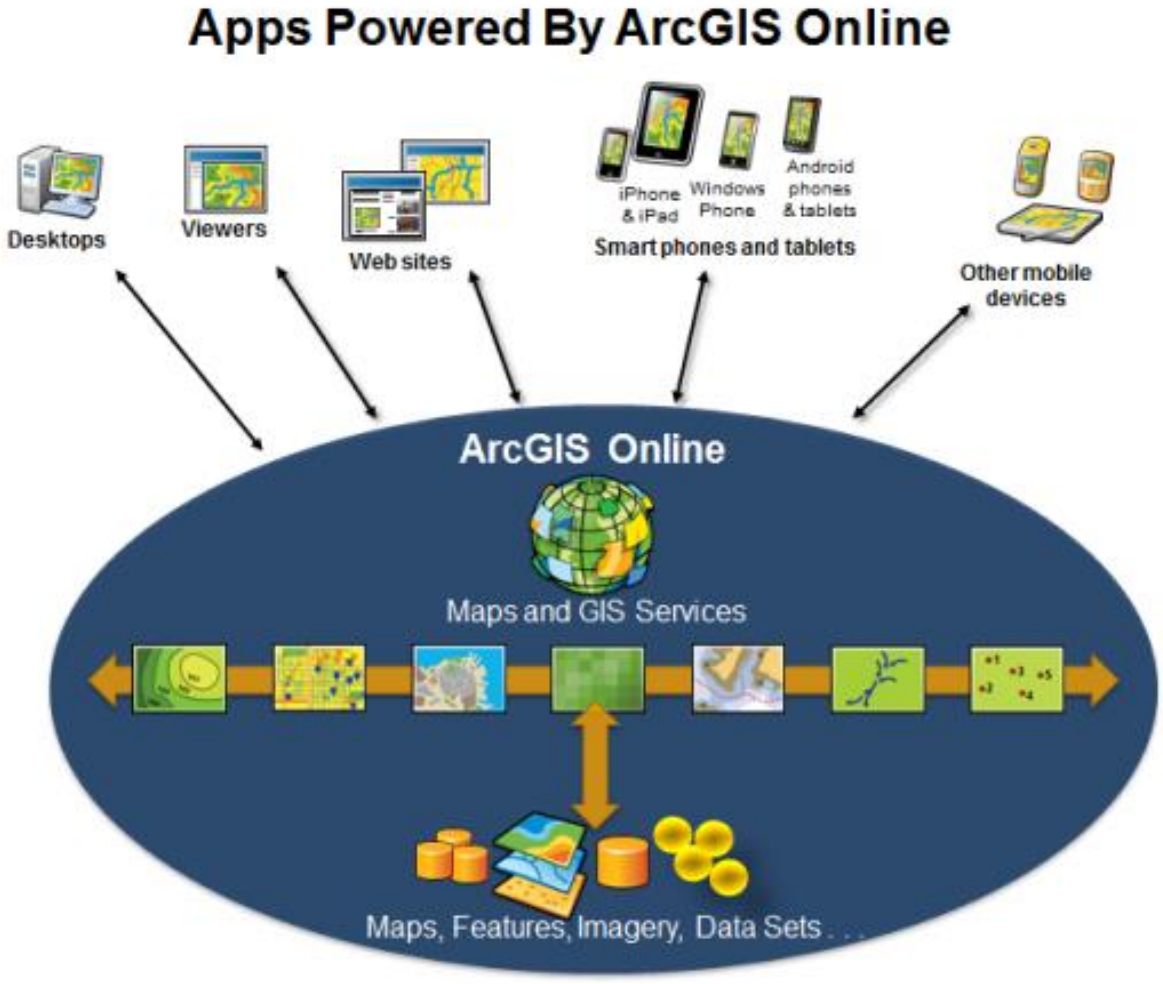

Рисунок 2.2 — Структура ArcGIS Online

Та серед мінусів можна зазначити такі пункти:

1. Важка у вивченні.
2. Увесь функціонал доводиться створювати власноруч.
3. Доцільність використання для автоматизації процесів під сумнівом.

4. Висока ціна для комерційного використання.

Взагалі ринок веб-застосунків, що працюють з геоінформаційними даними досить старий та багатий, але під вузьконаправлені або персональні задачі важко знайти готове рішення, що не потребуватиме тривалого процесу навчання по роботі з ним.

Також можна знайти велику кількість якихось незначних проектів, що не матимуть ніякої капіталізації та наукової цінності, що зайвий раз підтверджує факт складності розробки зручного продукту, яким віднайде популярність у кінцевого користувача.

2.5 Висновки до розділу

Розділ включає описання програмної області системи, її особливості та проблеми. Розглянуто існуючі рішення на ринку програмного забезпечення ГІС з визначенням проблемних місць в кожному з них.

3. ЗАСОБИ РЕАЛІЗАЦІЇ ПРОГРАМНОЇ СИСТЕМИ

Аналізуючи поставлену задачу та методи її вирішення, було вирішено розроблювати програмний комплекс на основі веб-технологій. Головною перевагою веб-застосунку перед іншими варіантами є його універсальність і можливість використання на будь-яких пристроях без портування на цільову операційну систему (браузер і його віртуальна машина виступає як цільова універсальна операційна система і комп'ютер).

Для створення веб-інтерфейсу було використано мови HTML5, CSS3, JavaScript. Для відображення карти та візуалізації даних на ній було використано бібліотеку Leaflet, що вільно розповсюджується й має відкритий вихідний код. Бібліотека Bootstrap використовувалась для створення адаптивної та зручної розмітки.

3.1 Середовище розробки веб-застосунків Visual Studio Code

Основним середовищем розробки було обране рішення від компанії Microsoft — Visual Studio Code. Компанія позиціонує його як "легкий" редактор коду для кроссплатформенной розробки веб- і хмарних додатків. Включає в себе відладчик, інструменти для роботи з Git, підсвічування синтаксису, IntelliSense і засоби для рефакторинга. Має широкі можливості для кастомізації: користувацькі теми, поєднання клавіш і файли конфігурації. Розповсюджується безкоштовно, розробляється як програмне забезпечення з відкритим вихідним кодом, але готові збірки розповсюджуються під проприетарной ліцензією.

Багато можливостей Visual Studio Code не доступні через графічний інтерфейс, найчастіше вони використовуються через палітру команд або JSON файли (наприклад, призначені для користувача настройки). Палітра команд представляє собою подобу командного рядка, яка викликається поєднанням клавіш.

Visual Studio Code був анонсований 29 квітня 2015 року компанією Microsoft на конференції Build, і незабаром була випущена бета-версія.

18 листопада 2015 року Visual Studio Code був випущений під ліцензією MIT, а вихідний код був опублікований на GitHub. Анонсована підтримка розширень.

14 квітня 2016 року Visual Studio Code вийшов зі стадії бета-тестування.

З 2018 року з'явилися розширення Python для Visual Studio Code з відкритим вихідним кодом. Воно надає розробникам широкі можливості для редагування, налагодження і тестування коду.

На березень 2019 року за допомогою вбудованого в продукт призначеного для користувача інтерфейсу можна завантажити і встановити кілька тисяч розширень тільки в категорії «programming languages» (мови програмування).

Для розробки даної системи було використано останню версію редактору — 1.34. Серед особливостей цього редактору коду можна відмітити такі пункти:

1. Безкоштовний текстовий редактор з відкритим вихідним кодом.
2. Має IntelliSense.
3. Невеликий розмір і вимоги до ОЗУ. З IntelliSense потрібно близько 300 МБ ОЗУ.
4. Працює на старих комп'ютерах. (Запуск і раніше йде повільно, особливо якщо замість CMD використовується PowerShell).
5. Більш низька підтримка (але наявність відкритого вихідного коду, дає вам змогу змінювати його самостійно).
6. В основному використовується для веб-розробки.
7. Велика база доступних розширень та плагінів, що зроблять вашу розробку ще швидшою.
8. Кросплатформений.
9. Має вбудований термінал (PowerShell працює занадто повільно при запуску).
10. Підсвітка синтаксису, в залежності від мови.
11. Управління системою контролю версій Git.
12. Користувацькі теми дають можливість обрати зручний варіант теми.

Для пришвидшення написання коду розмітки використовувалось розширення Emmet, що можна встановити як додатковий плагін в Visual Studio Code. [9]

Emmet (попередня назва — Zen Coding) — це набір плагінів для різних популярних текстових редакторів, що використовуються для швидкісного написання HTML і CSS коду. Синтаксис Emmet досить простий і не вимагає великих зусиль з боку розробника для його вивчення, в той же час використання цього інструменту дозволяє прискорити процес написання верстки в кілька разів.

Подивимося правді у вічі: написання HTML-коду вимагає часу, з усіма тегами, атрибутами, лапками, дужками і так далі. Звичайно, в більшості текстових редакторів є підказки, які сильно допомагають, але все одно доведеться багато друкувати. Emmet миттєво перетворює прості абревіатури в повноцінні блоки коду.

Для написання HTML в Emmet використовуються 12 типів селекторів:

1. # — створює атрибут id
2. . — створює атрибут class
3. []— створює будь-які інші атрибути, в тому числі і призначені для користувача
4. > — Робить перехід на один рівень нижче
5. + — створює сусідні елементи на тому ж рівні
6. ^ — робить перехід на рівень вгору
7. * — примножує елементи
8. $ — замінюється числом, кожен раз збільшується на одиницю
9. $$ — те ж саме, тільки двухзначне
10. {} — додає текстовий вміст елементів
11. ()— групує елементи
12. : — використовується для деяких елементів, таких як <input>, <a>, <link> і ін., І задає для них атрибути

Даний плагін доступний не лише у Visual Studio Code, але й у більшості сучасних текстових редакторів.

Visual Studio Code дозволяє реалізувати різні сценарії роботи і виходить за рамки звичайного редактора коду. Він має широкий вибір додаткових плагінів, що

значно спрощують та пришвидшують розробку веб-застосунків. Без сумніву, це один з найкращих редакторів коду, особливо для веб-розробника, на сьогоднішній день.

3.2 Основні технології, використані при розробці веб-інтерфейсу

Для створення клієнтської частини системи використовувався стандартний стек технологій для веб-розробки, а саме: HTML5, CSS3, JavaScript, а також додаткові бібліотеки JQuery та Bootstrap, які автоматизують створення інтерфейсу. Основою ж, для створення візуалізації геопросторових даних та їх відображення на карті, стала бібліотека Leaflet, що являє з себе безкоштовний та відкритий спосіб створення інтерактивних карт на будь-якому веб-сайті. Для надання користувачу можливості самому інтерактивно створювати об’єкти на карті використовується розширення цієї бібліотеки, створене користувачами під назвою Leaflet Draw, що надає інструменти для створення своїх геометричних об’єктів на карті, зберігаючи при цьому всю географічну інформацію по ним.

Основою розмітки веб-сторінки став останній стандарт мови розмітки гіпертекстових документів HTML5.

HTML5 — це стек програмних рішень, який визначає властивості та поведінку вмісту веб-сторінки, впроваджуючи в нього шаблон на основі розмітки.

HTML5 є п'ятою і поточною основною версією стандарту HTML. В даний час вона існує у двох стандартизованих формах: HTML 5.2 від W3C і HTML Living Standard від WHATWG (невеликий консорціум з чотирьох виробників браузерів), призначений в першу чергу для розробників браузерів. Існують незначні конфлікти між специфікаціями двох груп.

HTML5 був вперше випущений у відкритій формі 22 січня 2008 року з основним оновленням і статусом "Рекомендовано W3C" у жовтні 2014 року. Його метою є вдосконалення мови з підтримкою новітніх мультимедійних та інших нових можливостей. HTML5 призначений для включення не тільки HTML 4, але також HTML, XHTML 1 і DOM Level 2.

HTML 5 включає в себе детальні моделі обробки для підтримки більш взаємосумісних реалізацій; він розширює, вдосконалює і раціоналізує розмітку, доступну для документів, і вводить інтерфейси розмітки і прикладного програмування (API) для складних веб-додатків. З тих же причин, HTML5 також є кандидатом для крос-платформних мобільних додатків, оскільки він включає в себе функції, розроблені для малопотужних пристроїв.

Багато нових синтаксичних функцій включено. Для стандартного включення та обробки мультимедійного та графічного вмісту додано нові <video>, <audio> та <canvas> елементи, а також підтримка вмісту масштабованої векторної графіки (SVG) та MathML для математичних формул. Щоб збагатити семантичний зміст документів, додаються нові елементи структури сторінки, такі як <main>, <section>, <article>, <header>, <footer>, <aside>, <nav>, і <figure>.

API і об'єктна модель документа (DOM) тепер є фундаментальними частинами специфікації HTML5.[6]

Підтримкою для HTML є CSS, що забезпечує позиціонування та стилізацію елементів сторінки.

Каскадні таблиці стилів (CSS)[4] — це таблиця стилів, яка використовується для опису відображення документа, написаного на мові розмітки, наприклад HTML. CSS є частиною ядра технології World Wide Web, разом з HTML і JavaScript.]

CSS розроблений, щоб дозволити розділення відображення і вмісту, включаючи макет, кольори та шрифти. Цей поділ може поліпшити доступність контенту, забезпечити більшу гнучкість характеристик віжображення, дозволити декільком веб-сторінкам обмінюватися форматуванням, вказавши відповідний CSS в окремому файлі .css, і зменшити складність і повторення в структурному змісті.

Розділення стилів та вмісту також робить можливим представлення однієї сторінки розмітки в різних стилях для різних методів візуалізації.

CSS3 — це остання еволюційна зміна мови Cascading Style Sheets, і вона спрямована на розширення CSS2.1. Вона привносить давно очікувані нововведення, такі як закруглені кути, тіні, градієнти, переходи або анімація, а також нові макети, такі як multi-columns, “гумовий” дизайн або сітковий макет. Експериментальний

функціонал позначений спеціальними префіксами розробника, і він не повинен використовуватися у виробничому середовищі, або повинен використовуватися з особливою обережністю, так як його синтаксис і поведінка може бути змінена в майбутньому.

CSS Level 2 знадобилося 9 років, з серпня 2002 до червня 2011, щоб отримати статус рекомендації. Це сталося через те, що кілька вторинних особливостей затримували всю специфікацію. Щоб прискорити стандартизацію безпроблемних частин, Робоча група CSS W3C в своєму рішенні, відомому як “Пекінська доктрина”, розділила CSS на менші компоненти, звані модулями. Кожен з таких модулів тепер є незалежною частиною мови і проходить стандартизацію в своєму темпі, незалежно від інших частин. Коли одні модулі вже мають статус рекомендації W3C, інші все ще перебувають в стадії розробки. Так само з'являються нові модулі, коли в цьому є необхідність.

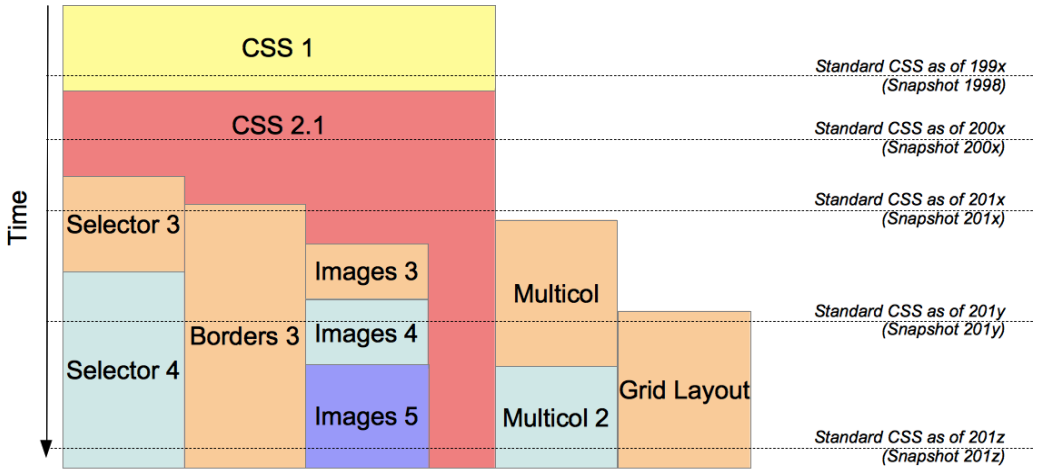

Рисунок 3.1 — Історія стандартів CSS

В CSS, селектори оголошують, яка частина розмітки стилю застосовується до відповідних тегів і атрибутів в самій розмітці.

Селектори можуть застосовуватися до наступного:

1. Всі елементи певного типу, наприклад заголовки другого рівня h2.
2. Елементи, визначені атрибутом, зокрема: id та class.
3. Елементи в залежності від того, як вони розміщені відносно інших у дереві документу.

Класи та ідентифікатори чутливі до регістру, починаються з букв і можуть включати буквено-цифрові символи, дефіси та підкреслення. Клас може застосовуватися до будь-якої кількості екземплярів будь-яких елементів. Ідентифікатор може бути застосований лише до одного елемента.

Взаємодія користувача з інтерфейсом відбувається за допомогою мови програмування JavaScript.

JavaScript (JS)[5] — динамічна, об’єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується як частина браузера, що надає можливість коду на стороні клієнта (такому, що виконується на пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

Цю мову можна зустріти буквально скрізь. Вона підтримується на всіх операційних системах, у будь-яких типах браузерів, на настільних комп'ютерах і на мобільних пристроях. Також, дуже важливо, щоб програми JavaScript працювали без встановлення їх на комп'ютерах користувачів. Насправді, вже важко згадати не такі давні часи, коли компанії, розгортаючи клієнт-серверні програми у своїх внутрішніх мережах, витрачали тижні з проблемами налаштування цих програм і з несподіваними помилками. Зіткнувшись з такими кошмарами, ви можете оцінити привабливість JavaScript.

JavaScript відомий тим, що не підтримує об'єктно-орієнтоване програмування. Фактично, розробники JavaScript традиційно обходять цей недолік, використовуючи всілякі дивні конструкції. Ці конструкції можуть означати щось для того, хто вже вивчав ООП (і навіть для того, хто ще не вивчав, а просто скопіював певний шаблон у свій код і звик працювати з ним).

Об'єктно-орієнтоване програмування (ООП) — це підхід до моделювання та організації коду. Якщо методи ООП застосовуються правильно, вони допомагають програмісту створювати простий і добре організований код. Крім того, ООП спрощує повторне використання важливих функціональних можливостей програми[12].

Мова JavaScript - це інструмент, що дозволяє застосовувати складні речі на веб-сторінці — кожен раз, коли щось відбувається на веб-сторінці, складніше, ніж просто її статичне відображення — відображення періодично-оновлюваного вмісту чи інтерактивні карти, або ж 2D / 3D анімацію, а може перемотка відео в плеєрі і т.д. — ви можете бути впевнені, що швидше за все — цього не було б без JavaScript. Це третій, найважливіший, шар багатошарового торту стеку веб-технологій, два з яких (HTML і CSS) було описано вище.

Мову JavaScript класифікують як прототипну (підмножина об’єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам’яттю, прототипне наслідування, функції як об’єкти першого класу.

JavaScript має ряд ознак, які притамані функціональним мовам — функції як об'єкти першого класу, об'єкти як списки, каррінг, анонімні функції, замикання — що надає мові додаткової гнучкості.

Більшість дій в JavaScript виконується з HTML-сторінкою. В JavaScript сторінка представлена у вигляді об'єктної моделі DOM (Document Object Model). Будь-які дії з сторінкою вимагають виклику відповідного методу DOM.

3.3 Технології для створення карти та візуалізації геопросторових даних, бібліотека Leaflet та її аналоги

Окрім основного стеку технологій веб-розробки, в проекті було використано допоміжні бібліотеки, щоб отримати можливість створити веб-відображення геопросторових даних. Можливість візуалізації даних на карті надає бібліотека для створення інтерактивних карт Leaflet

Leaflet — бібліотека з відкритим вихідним кодом, написана на JavaScript, призначена для відображення карт на веб-сайті. Підтримує більшість мобільних і стаціонарних платформ з числа тих, що підтримують HTML5 і CSS3.

На ряду з OpenLayers і Google Maps API — одна з найбільш популярних картографічних JavaScript-бібліотек, використовувана на таких великих сайтах, як Flickr, Foursquare, Craigslist, Data.gov, IGN, проектах Вікімедія, OpenStreetMap, Meetup, WSJ, MapBox, CloudMade, CartoDB та інших.

Leaflet дозволяє розробнику, не знайомому з ГІС, легко відображати растрові карти, що складаються з маленьких фрагментів — тайлів, з, можливо, додатковими шарами, накладуваних поверх основних. Шари можуть бути інтерактивними, наприклад, показувати підсказку при кліку по маркеру.

Leaflet підтримує шари Web Map Service (WMS), GeoJSON, векторні і тайлові шари. Багато інших типів шарів підтримуються додатковими модулями.

Як і в інших картографічних веб-бібліотеках, в Leaflet реалізована наступна модель: відображається базова карта з, можливо, растровими і векторними шарами, що накладаються поверх неї.

Основні типи об'єктів Leaflet:

1. Растрові типи (TileLayer і ImageOverlay).
2. Векторні типи (Path, Polygon і специфічні типи, такі як Circle).
3. Групові типи (LayerGroup, FeatureGroup і GeoJSON).
4. Керуючі елементи (Zoom, Layers і т.д.).

Окремо дана бібліотека не володіє вбудованими функціями та інструментами для створення власних геометричних об’єктів на карті за допомогою перетягування курсору, лише функціями для їх задання. Тож для вирішення цієї проблеми користувачама було створено додатковий плагін під назвою Leaflet Draw, що вміщує в собі набір готових інструментів для розширення даного функціоналу.

Вона дозволяє створити на карті панель інструментів, для створення маркерів, чотирикутників, кіл, ліній та користувацьких полігонів. Має низку подій, які надають можливість отримувати інформацію про, створенні користувачем, фігури[7].

Для малювання створюється окремий шар на карті, на якому й відбувається весь процес. Створені фігури можна редагувати, задавати стилі та всіляко налаштовувати.

У даного рішення є багато аналогів, таких як:

1. AmMap — бібліотека для побудови карт на Javascript, розроблена компанією amCharts. Вона володіє широким набором інструментів, та доволі таки обмеженим, та не підтримує глибокої кастомізації для створення свого потужного рішення.
2. AnyMap — ще одна популярна бібліотека для візуалізації даних від компанії AnyChart. Вона проста у використані, має можливість кастомізації, тому що вихідний код відкритий для редагування, але немає можливості створення своїх кастомних полів та об’єктів на карті, тож для вирішення задачі не підходить
3. Google GeoCharts — рішення від IT-гіганта Google, має набір стандартних методів та класів, але він доволі таки обмежений у функціональності, тож більше підходить для простих проектів, де потрібна здебільшого можливість розмальовувати області на карті, а вихідний код бібліотеки закритий для редагування.
4. Highmaps — розробник Highsoft створив цікаве рішення з відкритим кодом, що підтримує користувацькі рішення для додання в код, та функціонал для роботи з географічними даними дуже бідний. І до ще одного недоліку можна віднести неможливість роботи без бібліотеки jQuery, від якої останнім часом відмовляється більшість відомих IT-компаній через його громіздкість та розвиток нативного JavaScript.
5. jQuery Mapael — побудована на jQuery та Raphael, що робить її доволі таки важкої для підвантаження користувачем, а функціонал більше підходить для створення маршрутів на картах.
6. jVectorMap — розробкою займалась всього одна людина, тож продукт вийшов обмежений у можливостях, які не задовільнять рішення всіх проблем, що постають в проекті. Також підтримує лише GeoJSON формат сторонніх карт.
7. Kartograph — продукт від Gregor Aisch, дуже цікаве рішення, що розробив також один програміст без команди. Має масу візуальних ефектів для

кольорової візуалізації даних, навіть такі, що не має жодна інша, наприклад 3D-бари для відображення різного роду показників у вигляді стовпчиків прямо на самій карті.

Всі ці рішення безкоштовні, тож їх було детально вивчено, аби обрати серед них кращого, адже саме цей функціонал відповідає за основну задачу проекту, тож є найважливішим. На ринку, як виявилось в ході пошуку та аналізу, доволі таки багато різних бібліотек для задачі створення інтерактивних карт, та при більш детальному розгляді було встановлено, що більшість з них не володіє необхідними можливостями для вирішення поставленної задачі. Більшість з них потребує jQuery для роботи, або ж має закритий код на пару з обмеженим набором методів, що не дає права додати свої алгоритми. Leaflet ж в цьому змаганні вийшла беззаперечним переможцем. Через свої широкі можливості до модифікації, створення користувацьких плагінів і високим рівнем абстракції, вона заслужено стала найпопулярнішим рішенням для розв’язання задач, що пов’язані з географією та програмуванням карт.

3.4 Технології для взаємодії з сервером

Оскільки система складається з серверної частини та веб-інтерфейсу, виникає проблема їх зв’язку між собою. Отримати доступ до даних серверу та бази даних ми можемо за допомогою HTTP-запитів засобами JavaScript. Для цього була використана технологія асинхроного звернення до серверу під назвою AJAX.

AJAX (абревіатура від «Asynchronous Javascript And Xml») — технологія звернення до сервера без перезавантаження сторінки. За рахунок цього зменшується час відгуку і веб-додаток по інтерактивності більше нагадує десктоп.

Незважаючи на те, що в назві технології присутній буква X (від слова XML), використовувати XML зовсім не обов'язково. Під AJAX мають на увазі будь-яке спілкування з сервером без перезавантаження сторінки, організоване за допомогою JavaScript.

Технічно, за допомогою AJAX можна обмінюватися будь-якими даними з

сервером.

Зазвичай використовуються формати:

1. JSON — для відправки та отримання структурованих даних, об'єктів.
2. XML — якщо сервер чомусь працює в форматі XML, то можна використовувати і його, є засоби.
3. HTML / текст — можна і просто завантажити з сервера код HTML або текст для показу на сторінці.
4. Бінарні дані, файли — набагато рідше, в сучасних браузерах є зручні засоби для них.

Для обміну даними з сервером використовується спеціальний об'єкт XmlHttpRequest, який вміє відправляти запит і отримувати відповідь з сервера.

При використанні AJAX:

1. Користувач заходить на веб-сторінку і натискає на який-небудь її елемент.
2. Скріпт (на мові JavaScript) визначає, яка інформація необхідна для оновлення сторінки.
3. Браузер відправляє відповідний запит на сервер.
4. Сервер повертає лише ту частину документа, на яку прийшов запит.
5. Скрипт вносить зміни з урахуванням отриманої інформації (без повного перезавантаження сторінки).

Такий спосіб оновлення даних на сайті, без його перезавантаження робить будь-який веб-додаток зручнішим у використанні.

3.5 Засоби функціонування серверу

Серверна частина мікросервісної платформи була створена з використанням об'єктно-орієнтованої мови — Java.

Особливою рисою мови програмування Java можна виділити те, що її код на старті переводиться в унікальний байт-код, що абсолютно незалежить від пристрою. Після ж, байт-код інтерпретується в Java Virtual Machine (JVM). Тому, Java відрізняється від типових представників інтерпретованих мов, серед яких

PHP і Perl, програмний код котрих миттєво викликається інтерпретатором. Та все таки, Java є не просто компільованою, як C або C++.

Така архітектурна складова відкриває можливості крос-платформеності і апаратної портативності програм Java, так що програми на цій мові можна запускати на будь-яких платформах — будь-то Windows, Linux чи Mac OS — без жодної перекомпіляції. Для кожного пристрою чи платформи можна реалізувати віртуальну машину JVM.

Java являє з себе мову з C-подібним синтаксисом і схожа в цьому відношенні на C++ та C#. Так що, якщо вам доводилось працювати з однією з цих мов, вам буде простіше освоїти Java.

Також Java відрізняє від інших те, що вона включає автоматичний збір сміття. Це ж означає те, що вам не потрібно вручну випускати пам'ять з попередньо використовуваних об'єктів, як у C++, оскільки збирач сміття буде робити це автоматично для вас.

Java є об'єктно-орієнтованою мовою. Вона працює з поліморфізмом, успадкуванням, статичною типізацією. Об'єктно-орієнтована спрамованість мови дозволяє вирішувати завдання створення громіздких, але в той же час гнучких, логічних та розширюваних систем.

На сервері використовується також фреймворк Java — Spring Boot, що значно пришвидшує написання програмного коду та спрощує весь процес. Цей фреймворк надає можливість побудови зручного REST API для легкої взаємодії з веб-відображенням. Такий web-додаток легко підтримувати та редагувати, адже Spring має готовий набір рішень для більшості виникаючих проблем при побудові будь-якої системи.

Багато пакетів з налаштуваннями фреймоворку постають на вибір розробнику, що проектує майбутній застосунок, в наявності набори і для побудови веб-сторінок, для роботи з реляційними базами даних. Cloud пакет представляє бібліотеки взаємодії з різними хмарними технологіями. Ops — надає методи діагностики і тестування функій серверу, що значно спрощує пошук багів та розробку кінцевої системи[10].

3.6 Збереження даних

Дана система передбачає маніпуляції з великою кількістю даних користувача, які важливо надійно та зручно десь зберігати. У програмних продуктах для таких цілей використовуються різного роду сховища, що відносяться до сервеної частини системи. Надійним рішенням для цієї задачі буде створення власної бази даних на своїх носіях, щоб доступ до них мав лише клієнт, якому вони відповідають.

Будь-яка база даних потребує системи керування, аби інформацією в системі можна було вільно користуватись. Для цих цілей було обрано продукт PostgreSQL, що вперше булла випущена в 1996 році. Після довгої дороги розробки та покращення цей продукт добре зарекомендував себе на ринку.

СУБД PostgreSQL —система управління даними, що базується на мові запитів SQL і підтримує останні його стандарти. Вона необмежена по максимальному розміру бази даних, а розмір однієї таблиці дозволено навіть до 32 Тбайт.

Ця система має ряд основних переваг, серед яких:

1. Високопродуктивні та надійні засоби транзакцій та реплікацій.
2. Розширювана система вбудованих мов програмування, які можно розширювати офіційними модулями, або ж навіть створювати власні на мові C.
3. Підтримує наслідування.
4. Легко розширюється та модифікується.

Також варто відмітити, що PostgreSQL підтримує одночасну модифікацію бази даних двома або більше користувачами за допомогою технології Multiversion Concurrency Control (MVCC), що практично позбаляє необхідності створювати блокування зчитування.

Серед підтримуваних типів:

1. Чисельні типи (цілі, з фіксованою та плаваючою точкою та грошовий тип для представлення валют).

2. Символьні типи будь-якої довжини.
3. Двійкові типи.
4. Дату/час з великою варіативністю.
5. Булевий тип.
6. Геометричні примітиви.
7. Мережеві типи, такі як MAC-адреса та інші.
8. JSON.
9. Псевдотипи.

Підключення цієї бази даних вирішує проблему зберігання даних та їх отримання з серверу веб-інтерфейсом для подальшого відображення.

3.7 Висновки до розділу

Було розглянуто всі програмні засоби, що було використано в ході розробки програмного продукту з їх детальним описанням. Проведено аналіз аналогів та визначення кращого варіанту в даній предметній області.

4. ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Розробка веб-інтерфейсу програмного продукту поділялась на такі основні етапи:

1. Створення класів та функцій для отримання та обробки даних з серверу засобами JavaScript.
2. Підготовка оброблених даних та їх представлення на веб-сайті за допомогою HTML5 та CSS3
3. Модуль для оновлення даних в базі даних, в залежності від дій користувача в системі.
4. Інтеграція з іншими мікросервісами платформи.

Т

4.1 Структура бази даних

База даних, що побудована на PostgreSQL має чітку структуру, що дозволяє зберігати дані відповідно до їх типу. Вона містить дві основних таблиці:

— Fields — список полів користувача.

— Measurements — виміри пенетрометра.

Колекція Fields включає в себе наступні поля:

1. id — унікальний ідентифікатор.
2. name — ім’я поля: текст.
3. coordinates — координати поля: текст.
4. enabled — працездатність поля: логічний тип.
5. createdAt — дата створення: дата.
6. description — опис поля: текст.

Колекція Measurements:

1. id — унікальний ідентифікатор.
2. densities — значення виміру: текст.
3. coordinates — координати місця виміру: текст.
4. type — тип накінечника: текст.

5. createdAt — дата виміру: дата.

Більшість даних представлені колекціями об’єктів географічного типу, тож простіше всього представляти їх у вигляді звичайної строки, яку потім приводити до необхідного типу за допомогою регулярних виразів.

4.2 Архітектура системи в цілому

Для побудови системи було обрано сервісно-орієнтовну модель — мікросервісної платформи. Основна ідея якої заключається в розбитті громіздкого проекту на якомога менші частини, що відповідатимуть за певну частину функціоналу, максимально не залежачи один від одної. Ці елементи й носять назву мікросервісів.

Замість важких протоколів для взаємозв’язку між частинами таких систем , як наприклад — SOAP, використовуються більш легші та швидші, наприклад, REST з використанням JSON файлів для обміну необхідною інформацією

При використанні такого типу архітектури проект набуває більшої зв’язності та зменшує рівень зчепленності між робочими модулями за рахунок підвищення їх гранулярності [13](рисунок 4.1)

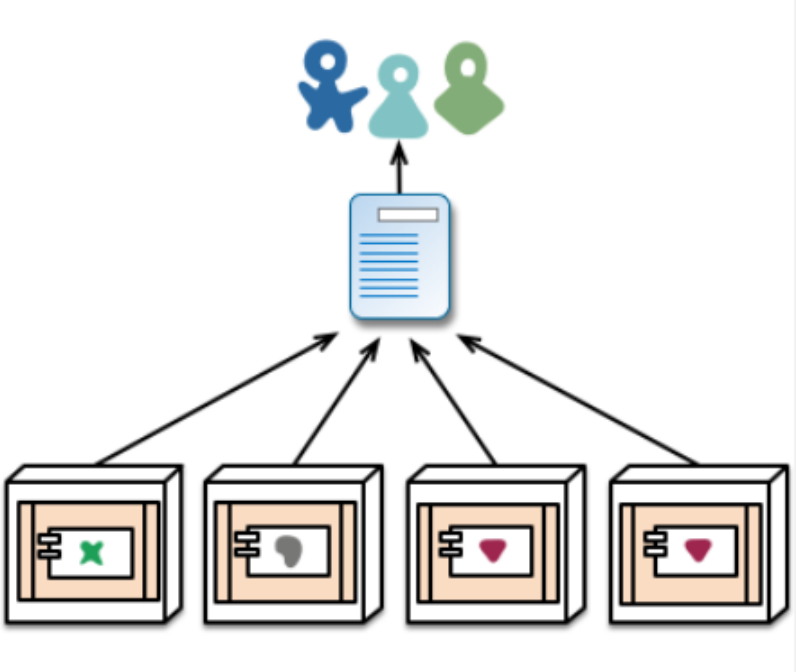

Рисунок 4.1 — Схематичне зображення архітектури мікросервісної платформи

Формально такий тип структури описати проблематично, тому що немає чіткого визначення як має бути побудовано кожен модуль. Та суть ж полягає у створенні модулів, що виконуватимуть мінімально-визначений функціонал.

В додаток до гнучкості та можливості розширення сервісу, кожен мікросервіс отримує чіткі фізичні межі, які дозволяють різним модулям бути написаними на різних мовах програмування та навіть різними командами. Кожен такий елемент системи може використовуватись окремо від даної платформи, будучи незалежним програмним продуктом.

В даному проекті цей принцип був застосований для можливості подальшого розширення продукту необхідними для клієнта функціями. Звязок між усіма серверними сервісами відвувається за допомогою веб-інтерфейсу, що може отримувати дані за допомогою HTTP протоколу передачі даних.

Для зручного зв'язку всі сервіси будуються на принципах REST, тобто створено REST API, особливістю якого є те, що сервер не запам’ятовує стан користувача між кожним із запитів, в кожному окремому передається необхідна цільова інформація та якийсь ідентифікатор, наприклад token-ключ, що отримується клієнтом після вдалої авторизації до веб-системи.

Тобто вся взаємодія з сервером зводиться до 4 основних дій:

1. Отримання інформації з сервера, який в свою чергу бере її з бази даних програми.
2. Додавання нових елементів.
3. Редагування існуючої на сервері інформації.
4. Видалення непотрібних даних з сховища.

За кожну операцію відповідає окремий HTTP-метод:

1. GET — отримує.
2. POST — відправляє.
3. PUT — модифікує.
4. DELETE — видаляє.

4.3 Класи сутностей та їх властивості

Основні структурні логічні одиниці веб-додатку це окремі JavaScript файли архітектурних модулів. В структурі проекту виділено такі основні частини: Fields (поля), Measurements (виміри), View (маніпуляції з DOM), Map (карта) Draw (відображення даних на мапі), Server (взаємодія з сервером та базою даних засобами AJAX).

— Fields — модуль, що включає в себе клас Field, який має відповідні поля, що реалізують властивості сутності: name(назва поля), coordinates(координати поля), createdAt(дата створення), description(короткий опис особливостей області), enabled(дієздатність поля), id(унікальний ідентифікатор з бази даних), polygon(об’єкт класу бібліотеки Leaflet — Polygon, що представляє з себе поле безпосередньо на карті), measurementsIn (значення властивості відповідає вимірам, що відбувались на території цього поля, вираховується за географічним алгоритмом пошуку координати в області). Окрім цього включає в себе метод приведення текстових координат з бази даних до об’єктів класу LatLng бібліотеки Leaflet, що являються колекцією з двох елементів для довготи і широти точки. Окрім цього є ряд методів для валідації, приведення типів, відображення та обробки даних, що стосуються полів та їх візуалізації на карті.

— Measurements — відповідає за виміри пенетрометра, й містить в собі одноімений клас для моделювання кожного виміру в межах веб-додатку. Має схожий набір методів до класу Fields для обробки даних, приведення типів та виводу інформації на карту.

— View — забезпечує навігацію по застосунку, відображення та зміну елементів DOM. В основному тут зберігається код, що відповідає за додавання слухачів подій для елементів веб-сторінки.

— Draw — відбувається ініціалізація всіх класів та методів для роботи з мапою, відповідає за створення, редагування та збереження даних з карти. відображає дані класів Field та Measurements на карті та забезпечує створення нових об’єктів за допомогою візуальних інструментів бібліотеки Leaflet Draw. Також

відповідає за генерацію карти, додання на неї тайлів карт та задання значень за замовчуванням для різних інструментів.

— Server — відповідає за обмін інформації з сервером, вміщує для цієї цілі набор функцій, що базуються на технології AJAX для динамічного оновлення станів, що дозволяє користувачу без затримок отримувати свою інформацію.

4.4 Архітектура веб-відображення

Кінцевий клієнт сервісу все одно бачитиме лише зовнішню картину системи, тобто її веб-інтерфейс, котрий базується на мові розмітки HTML5 та каскадних таблицях стилів — CSS3. За їх зміну відповідає нативний JavaScript без використання різного роду бібліотек та фреймворків. Для покращення естетичного враження від користування системо на сайті підключено Google Fonts та іконки для дизайну різних елементів з Font Awesome.

В першу чергу на сторінці відображається карта, яку повертає бібліотека Leaflet, що складається з тайлів, які будують мапу світу на основі відкритих даних проекту OpenStreetMap. Для неї задаються початкові координати відображення, тобто до якої локації скерується поле огляду користувача одразу після завантаження і з яким приближенням (рисунок 4.2)

```
var mymap = L.map('mapid', {
    center: [50.479511, 30.495351],
    zoom: 10,
    zoomControl: true,
});
```

Рисунок 4.2 — Ініціалізація об’єкту карти

Накладання тайлів на мапу потребує окремого шару, що включатиме їх в себе, такий клас в бібліотеці відповідно називається tileLayer, що приймає в конструктор посилання на дані в необхідному форматі.

Далі створюється об’єкт featureGroup — шар-контейнер для всіх елементів, що відображатимуться на карті з бази даних. В ньому ж і будуть створюватись поля, які

задасть користувач самостійно.

На всі необхідні елементи керування навішуються слухачі подій, й далі система очікує на дії користувача.

4.5 Динамічне представлення геопросторових даних

Веб-інтерфейс для візуалізації геопросторових даних складається з декількох частин, логіка кожної з яких описується в окремому JavaScript файлі для кращої структуризації проекту. Головною частиною кабінету користувача буде карта з даними, що є основою для візуалізації користувацьких даних.

Дані для веб-представлення отримуються з двох основних мікросервісів системи: сервіс звітів та сервіс полів та вимірів. Користувач, зайшовши до свого кабінету має змогу отримати всі можливі виміри по його підприємству за певний проміжок часу. Вони відображуться на мапі у вигляді маркерів у межах поля, яке було обрано (рисунок 4.3).

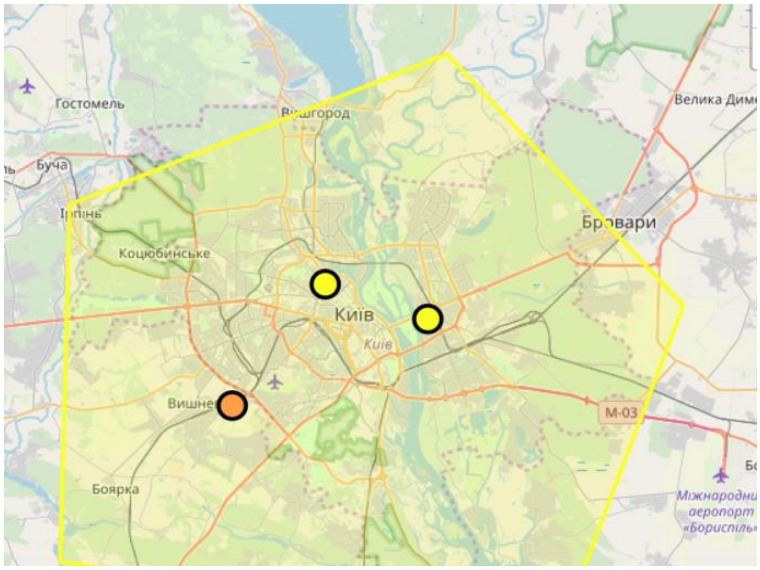

Рисунок 4.3 — Маркери у місцях вимірів на карті

З сервісу полів та вимірів отримується список, заданих користувачем, територій, що належать до його господарства з усією відповідною для них інформацією

JSON-дані, отримані з серверу десеріалізуються, оброблюються, на їх основі

створюються екземпляри класів Fields та Measurements після чого дані додаються до інтерфейсу систему. На виході ми отримаємо список полів (рисунок 4.4), вибравши одне з них ми отримаємо коротку інформацію по ньому, а на карті будуть відображені його межі по координатам

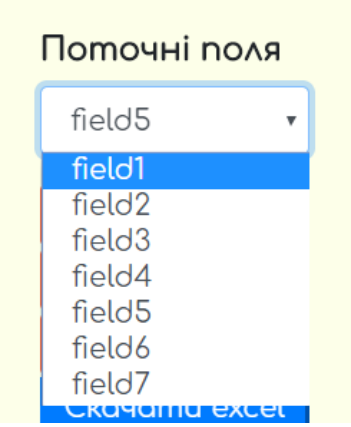

Рисунок 4.4 — Список полів

Також варто зауважити, що для користувача доступна можливість вибору типу карти для відображення: супутникова (рисунок 4.5) та звичайна (рисунок 4.6). Дані супутникової карти отримуються з серверів Google, для звичайної було обрано OpenStreetMap. В меню управління є змога приховати шар користувацьких відображень, а потім повернути його назад.

Рисунок 4.5 — Супутникова карта с полем

При переключені між типами карт не зникають жодні елементи на карті, що було задано користувачем. Режим супутникового відображення дає користувачеві можливість точніше розгледіти особливості рел’єфу та межі необхідних територій, для задання на карті.

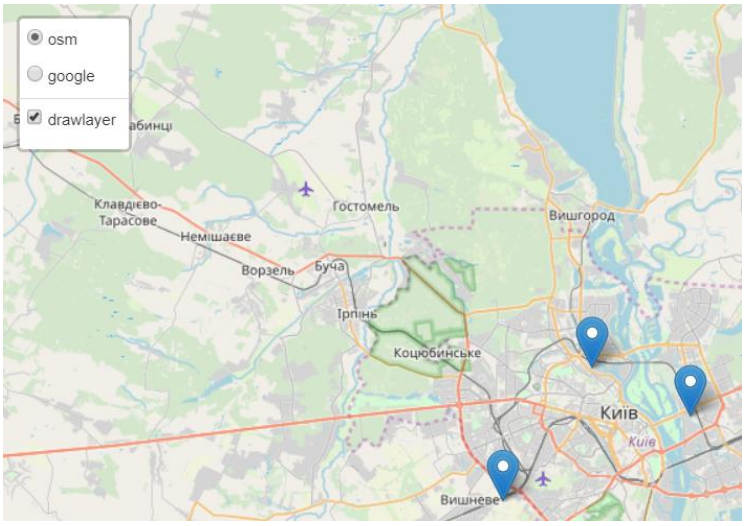

Рисунок 4.6 — Звичайна карта з інструментом вибору шарів

На шар мапи було навішано слухача події створення чи редагування об’єктів, який виконує функцію перехоплення відредагованого об’єкту, який одразу ж відправляється на сервер, звідки зберігається в базу даних.

4.6 Висновок до розділу

У розділі було подано інформацію про архітерну складову системи та розглянуто особливості кожного процесу більш детально з наглядною демонстрацією у формі скріншотів системи.

5. МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Розроблено програмний комплекс розроблений з використанням веб-технологій і тому працює в браузерах, які підтримують актуальні веб-стандарти. Додаткових встановлень, окрім авторизації не вимагає, система легка та адаптивна під різні пристрої.

5.1 Інсталяція та системні вимоги

Робота з системою не потребує ніякої інсталяцяї чи додаткового програмного середовища, окрім сучасного браузеру. Бажано мати стабільний інтернет та комп'ютер початкової потужності для запуску веб-переглядача, що підтримуватиме актуальні веб-технології.

5.2 Сценарій роботи користувача з системою

Веб-застосунок візуалізації даних містить у собі лише одного актора, що являє з себе власне користувача системи. Адміністратор ж не представлений окремим актором в системі, тому що немає необхідності постійного втручання до роботи платформи, вона повністю самостійна та самообслуговувана.

Клієнт платформи має спектр дій для виконання у веб-інтерфейсі. Кожна з яких включає варітивність для покращення User Expirience від користування системою. Наприклад при задані поля можна обрати один із декількох способів введення інформації до застосунку для різноманітно можливих даних, усі з яких проходять валідацію.

Діаграма прецедентів лаконічно описує дії та можливості клієнта у системі з усіма витікаючими наслідками та можливостями, extend стркілки вказують на

опціональність виконання того чи іншого варіанта(рисунок 5.1).

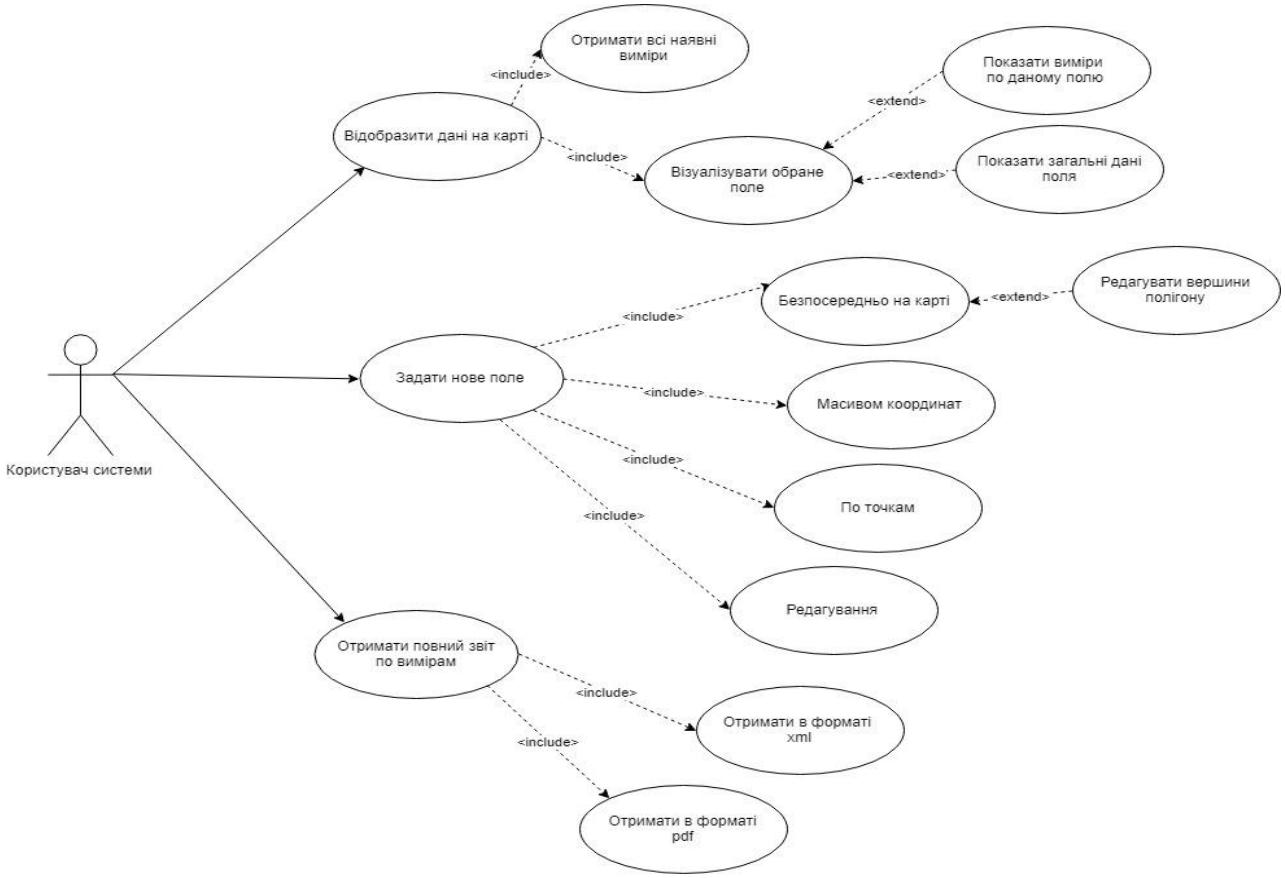

Рисунок 5.1 — Діаграма прецедентів системи

5.3 Інструкція з використання програмного продукту

При вході до веб-інтерфейсу користувач має змогу завантажити з серверу всі свої виміри, якщо вони там наявні. Або ж задати початкові поля на карті, для їх подальшого використання. Задати їх можна двома способами: задавши границі на карті, або ж точно визначити за допомогою масиву координат, який можна передати одразу масивом точок, або ж поточково (рисунок 5.2). Також необхідно надати назву полю, а опис вписується по бажанню.

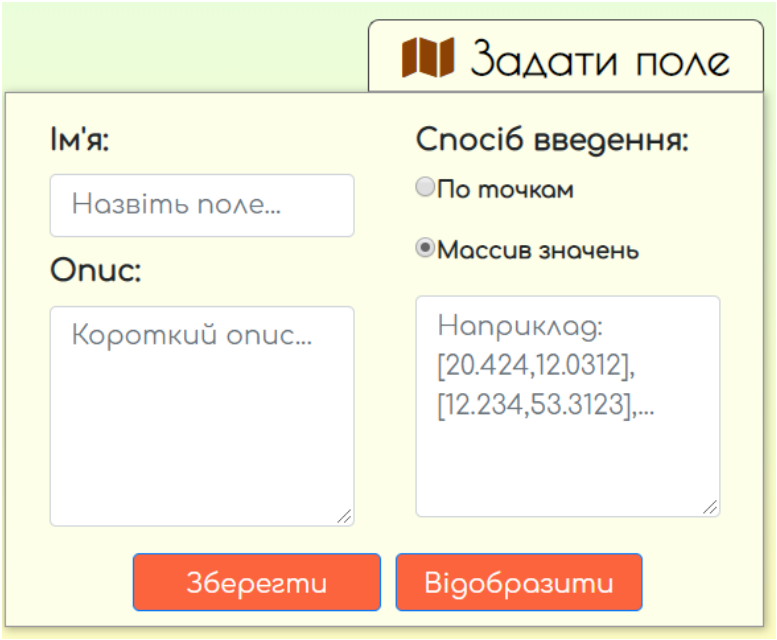

Рисунок 5.2 — Форма задання поля по координатам

На карті є елементи керування для створення полігонів за допомогою маніпуляцій з комп'ютерною мишею, вони знаходяться у правому верхньому куту мапи (рисунок 5.3). Також там знахоться іконки редагування створених полів та видалення, в разу необхідності.

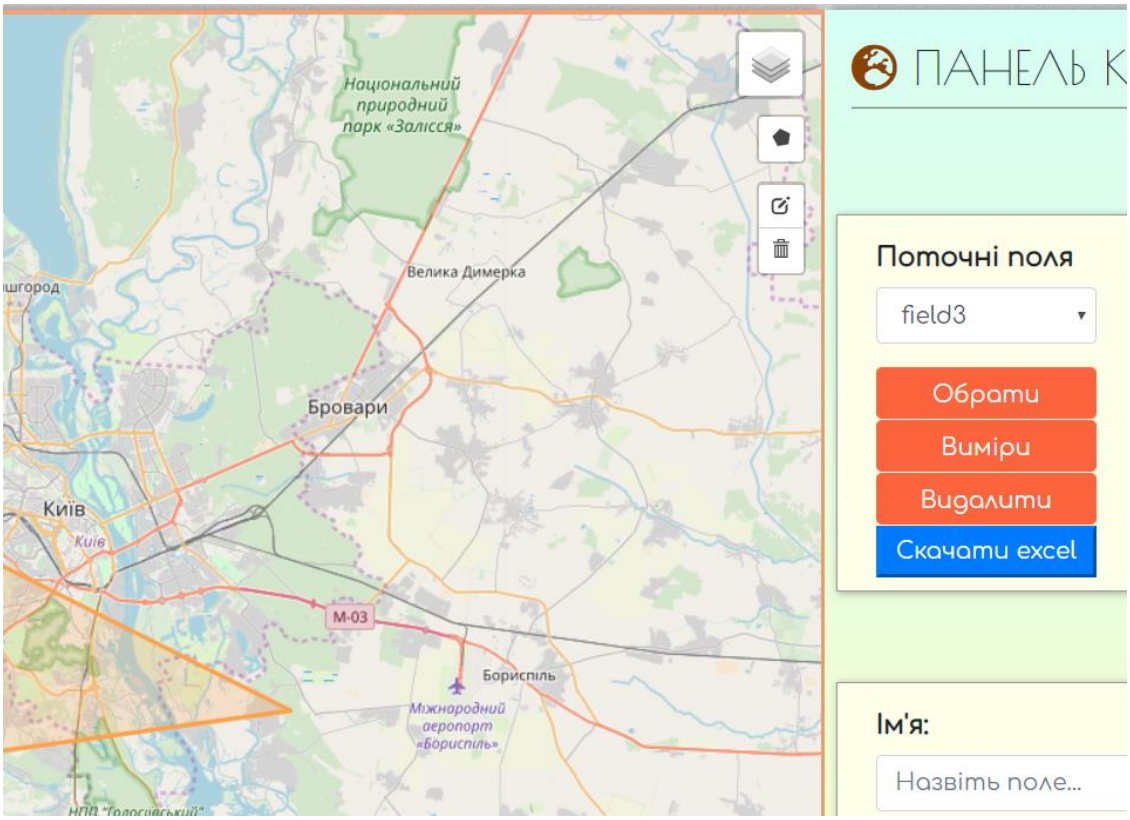

Рисунок 5.3 — Елементи контролю для створення полігонів

Задавши нове поле чи обравши існуюче, можна перевірити чи є в базі даних виміри, що потрапляють в межі заданої території, для цього необхідно натиснути кнопку “Виміри” (рисунок 5.4).

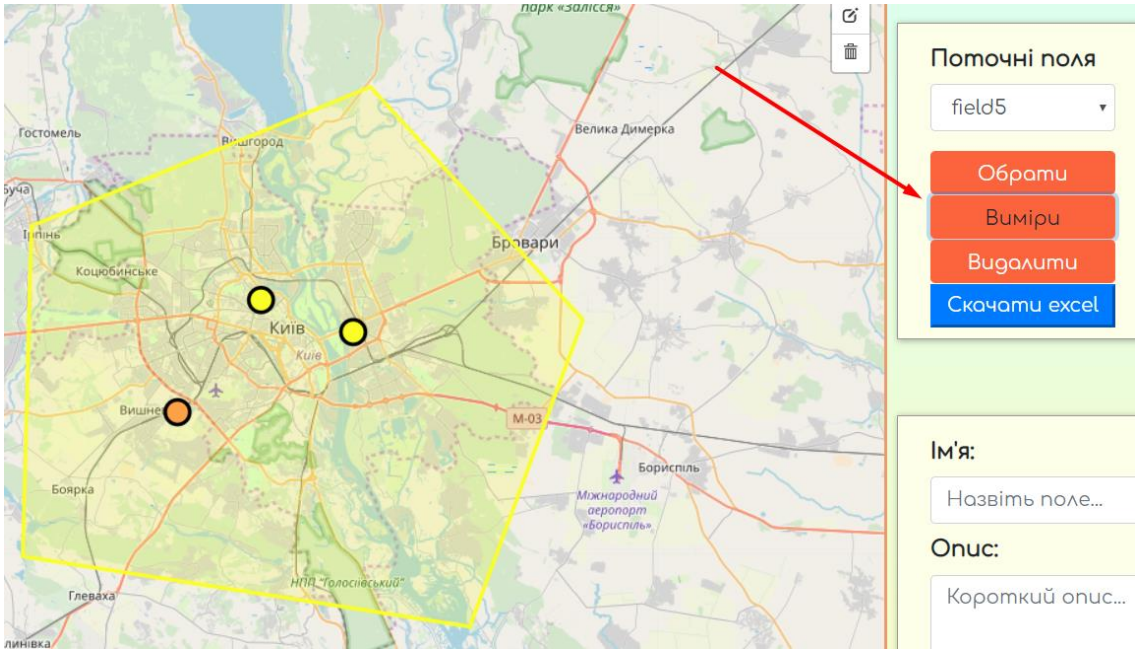

Рисунок 5.4 — Елементи контролю для створення полігонів

У разі відсутності вимірів в цій області, інтерфейс повідомить користувача про це модальним вікном із затемненням (рисунок 5.5).

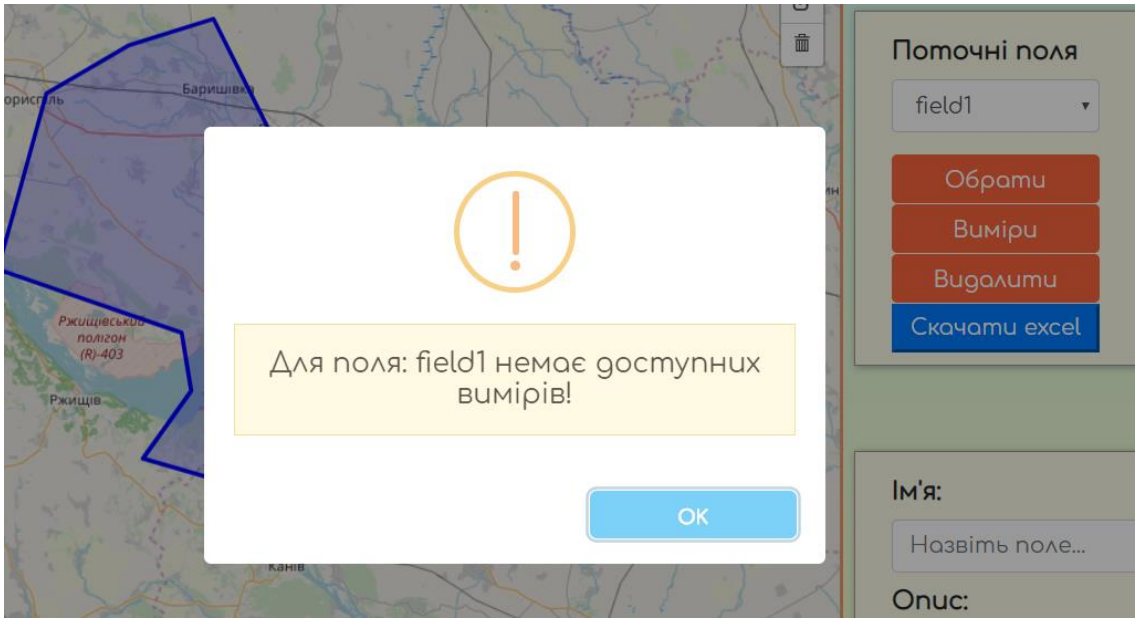

Рисунок 5.5 — Повідомлення про відсутність вимірів.

Також для користувача доступна можливість для редагування своїх існуючих полів, за допомогою інструмента “Edit”, на полі з’являються всі вершини та їх можна перетягувати за допомогою курсора мишки (рисунок 5.6).

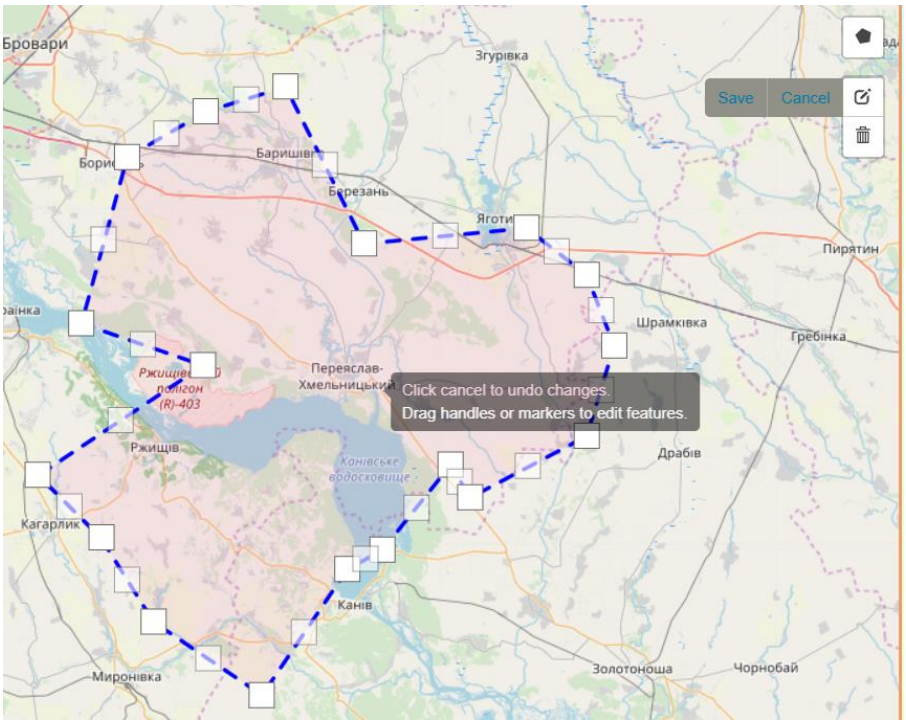

Рисунок 5.5 — Повідомлення про відсутність вимірів.

Отримування інформації про поля зроблені дуже зручним методом — досить лише клікнути в області поля для відкриття інформаційного вспливаючого вікна з усіма даними (рисунок 5.6).

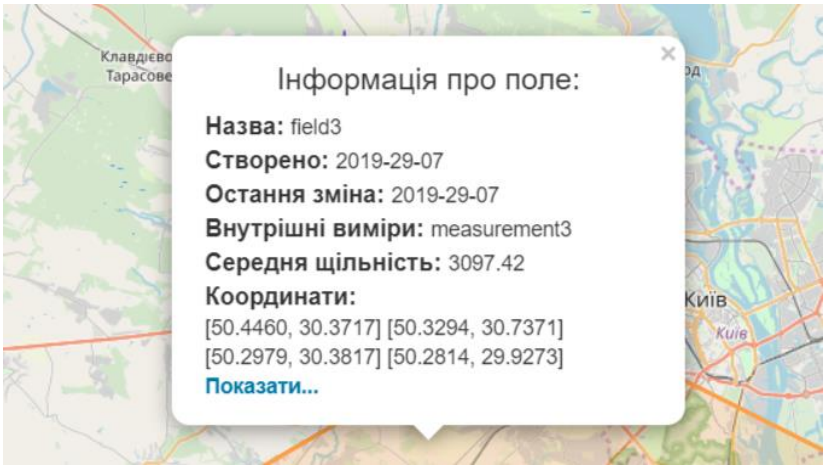

Рисунок 5.6 — Інформаційне вікно для поля.

Такий ж принцип використано для візуалізації даних для кожного виміру. Клікнувши по маркеру виміра, відкриється вспливаюче вікно (рисунок 5.7)

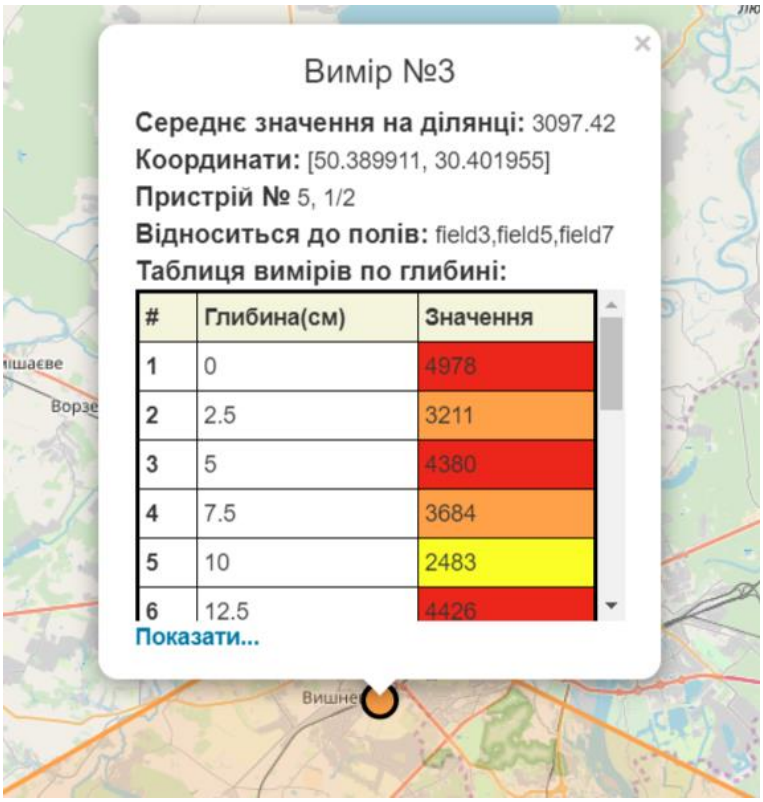

Рисунок 5.7 — Інформаційне вікно для виміру

По завершенню сесії користувачем, всі створенні дані передаються на сервер, що записує оновлення до бази даних.

5.4 Висновки до розділу

В розділі описано вимоги для запуску програмного забезпечення, які є мінімальними, коротка інструкція по інтерфейсу веб-застосунку з детальними скріншотами, що ілюструють систему.

ВИСНОВКИ

У ході виконання диплома розроблена унікальна веб-система для візуалізації геопросторових даних на інтерактивній карті з динамічним оновленням даних без затримок..

Система включає в себе серверну частину, що виконує зв’язок з базою даних, яка зберігає в собі інформацію про всі поля та виміри поточного користувача сервісу.

Дана система відповідає списку поставлених задач:

1. Створена карта для відображення геоінформаційних даних.
2. Написано модуль для взаємодії з сервером та базою даних за допомогою технології REST API.
3. Надано можливість створювати власні області-поля на карті веб-сайту.
4. Дані вимірів пенетрометру зручно візуалізуються у вспливаючому вікні.
5. Створено алгоритм для визначення приналежності виміру до обраної ділянки.
6. Користувацькі поля зафарбовуються у певний колір, в залежності від значень середніх вимірів всередині них.
7. Веб-сайт розроблено за принципами адаптивності, що надає змогу зручно користуватись ним, навіть за допомогою смартфону.
8. Інтерфейс включає можливість редагування існуючих полів з одночасним оновленням значень в базі даних.

Кінцева платформа готова до розширення функціоналу за рахунок мікросервісної архітектури, що зменшує рівень зчепленості різних модулів системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Конкурс стартапів Sikorsky Challenge [Електронний ресурс] – Режим доступу до ресурсу: https://www.sikorskychallenge.com.

2. Agro API [Електронний ресурс] – Режим доступу до ресурсу: https://agromonitoring.com/.

3. HTTP [Електронний ресурс] – Режим доступу до ресурсу: https://developer.mozilla.org/ru/docs/Web/HTTP.

4. Cascading Style Sheets (CSS) Standarts & Drafts [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3.org/Style/CSS/.

5. Флэнаган Д. JavaScript. Подробное руководство [Электронный ресурс] / Д. Флэнаган. — 2012. — С. 524–550. — Режим доступа: http://www.books.ru/books/javascript-podrobnoe-rukovodstvo-6-e-izdanie-1814274/ .

6. W3C Standerds HTML & CSS [Electronic resource]. — Access mode: https://www.w3.org/standards/webdesign/htmlcss.

7. Leaflet Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://leafletjs.com/reference-1.5.0.html.

8. Bootstrap 4 documentation [Електронний ресурс] – Режим доступу до ресурсу: https://getbootstrap.com/docs/4.3/getting-started/introduction/.

9. Visual Studio Code [Електронний ресурс] — Режим доступу: https://code.visualstudio.com/

10. Spring Boot [Електронний ресурс] — Режим доступу: https://spring.io/projects/spring-boot/

11. НАЦІОНАЛЬНИЙ СТАНДАРТ УКРАЇНИ. ҐРУНТИ. Визначення щільності ґрунтів методом заміщення об'єму [Електронний ресурс] – Режим доступу до ресурсу: http://ksv.do.am/GOST/DSTY_ALL/DSTY4/dsty_b_v.2.1-21-2009.pdf.

12. Riel A. J. Object-oriented design heuristics / Arthur J. Riel., 2019. – 231 с.

13. Ээлес П. The Process of Software Architecting / П. Ээлес, П. Криппс., 2009. – 432 с.

14. Таненбаум Э. Компьютерные сети / Эндрю Таненбаум., 2013. – 960 с. – (5).

15. Сиротинська А. П. Інформаційні системи підприємств малого бізнесу / Алла Павлівна Сиротинська. – Київ, 2008. – 264 с.

16. Цытович Н. Механика грунтов. Краткий курс / Николай Цытович., 2014. – 288 с.

ДОДАТОК 1

Веб-інтерфейс візуалізації геопросторових даних для мікросервісної платформи

Специфікація

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ51148_19Б

Аркушів 2

Київ – 2019

Позначення	Найменування	Примітки
Документація		
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТВ51148_19Б	Записка.doc	Пояснювальна записка
Компоненти		
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТВ51148_19Б	index.html css/style.css	Веб-інтерфейс системи
УКР.НТУУ" КПІ ім. Ігоря Сікорського"_ТЕФ_АПЕПС_ТВ51148_19Б	js/draw.js js/map.js js/field.js js/view.js js/measurements.js js/ leaflet.js	Модулі клієнської частини розробленої системи

ДОДАТОК 2

Модуль обробки та відображення даних користувацьких полів з серверу

Лістинг програмного модулю

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ51148_19Б 12-1

Аркушів 5

Київ – 2019

```
class Field {

    constructor(name, coordinates, createdAt, changedAt, description = "Це поле", square = 0, enabled, id = 0) {
        if(!isFieldExist(name)){
          this.name = name;
        }else {
            alert("Поле з таким іменем вже існує!");
            return false;
        }

        this.coordinates = Field.stringToArrayOfPoints(coordinates);
        this.createdAt = createdAt;
        this.changedAt = changedAt;
        this.description = description;
        this.enabled = enabled;
        this.square = square;
        this.id = id;
        this.polygon = L.polygon(this.coordinates,{attribution:this.name});
        this.measurementsIn = this.checkMeasurements();
    }

    checkMeasurements(){
        var intersected = new Array();
        for(var meas in measurements){
            if(this.name == measurements[meas].fieldName){
                intersected.push(meas)
            }else{
                if(isMarkerInsidePolygon(measurements[meas].marker, this.polygon)) {
                    console.log(meas, this.name, this.id);
                    measurements[meas].fieldsInfo(this.name,this.id);
                    intersected.push(meas)
                };


            }
        }
        return intersected;
    }

    showMeasurements(){
        if(this.measurementsIn.length==0){
        swal({
                text: "Для поля: " + this.name + " немає доступних вимірів!",
                icon: "warning",
                timer: 3000,
             });
        }
        for(let i=0; i < this.measurementsIn.length; i++){
            measurements[this.measurementsIn[i]].drawMarker();
        }

    }

    drawPolygon(){
        drawingLayer.clearLayers();
        var p_color = chooseColor(this.avrgMeasurements)
        let infoField = L.popup({
                closeButton: true,
                closeOnClick: false,
                maxWidth: '400px'
            })
            .setContent(this.fieldHtml);
        this.polygon.bindPopup(infoField);
```

```
        this.polygon.setStyle({color: p_color});
        drawingLayer.addLayer(this.polygon);

        mymap.fitBounds(this.polygon.getBounds());
    }

    get fieldHtml(){
        return `<div class='popup-info'>
                        <h5 style='text-align:center;'>Інформація про поле:</h5>
                        <span class="popup-line-name">Назва: </span>
                        <span> ${this.name} </span><br>
                        <span class="popup-line-name">Створено: </span>
                        <span> ${this.createdAt} </span><br>
                        <span class="popup-line-name">Остання зміна: </span>
                        <span> ${this.changedAt}</span><br>
                        <span class="popup-line-name">Внутрішні виміри: </span>
                        <span> ${this.measurementsIn}</span><br>
                        <span class="popup-line-name">Середня щільність: </span>
                        <span> ${this.avrgMeasurements}</span><br>
                        <span class="popup-line-name">Координати: </span><br>
                        <span class="collapse multi-collapse" id="more">${this.coordinatesString}</span>
                        <span><a href="#more" aria-expanded="false" class="font-weight-bold" data-
toggle="collapse">Показати...</a> </span><br></div>`;

    }

    static stringToArrayOfPoints(geo) {
        if (geo[0][0].lat) {
            console.log("array is beauty!");
            return geo[0];
        }
        var linesPin = geo.toString().split(",");
        if (linesPin.length % 2 != 0) {
            return false;
        }
        var linesLat = new Array();
        var linesLng = new Array();
        for (let i = 0; i < linesPin.length; i++) {
            if (i % 2) {
                linesLng.push(linesPin[i]);
            } else {
                linesLat.push(linesPin[i]);
            }
        }
        var latLngLine = new Array();

        for (let i = 0; i < linesLng.length; i++) {
            latLngLine.push(L.latLng(linesLat[i], linesLng[i]));
        }
        return latLngLine;
    }

    get coordinatesString() {
        let coordString = "";
        this.coordinates.forEach(function(element) {
            coordString += ("[" + element.lat.toFixed(4) + ", " + element.lng.toFixed(4) + "] ");
        });
        return coordString;
    }

    get avrgMeasurements(){
        var avrg = 0;
```

```
        this.measurementsIn.forEach(function(currentValue) {
            avrg += +measurements[currentValue].avrgDensities;
        });
        if(!avrg) return "Немає доступних вимірів"
        return avrg/this.measurementsIn.length;
    }


}

function chooseColor(dents){
    console.log(dents)
    switch(true){
    case (dents >= 0 && dents < 950) : return "#D1FFCD";
    case (dents >= 950 && dents < 1900) : return "#24FF28";
    case (dents >= 1900 && dents < 2900) : return "#FBFF27";
    case (dents >= 2900 && dents < 4000) : return "#FFA148";
    case (dents >= 4000 && dents < 6000) : return "#ED261B";
    default : return "blue";
    }
}


function isFieldExist(name){
    if (fields){
        for (var f in fields){
            if (f == name){
                alert("Поле с таким ім'ям уже існує. Задайте інше!");
                return true;
            }
        }
    }else return false;

}




function updateFieldsList() {
    var htmlFieldsList = document.getElementById("fieldsList");
    htmlFieldsList.innerHTML = ""
    for (var field in fields) {
        // console.log(fields[field]);
        let elem = document.createElement("option");
        elem.setAttribute("value", fields[field].name);
        elem.textContent = fields[field].name;
        elem.label = fields[field].name;
        htmlFieldsList.appendChild(elem);
    }

    var htmlMeasurementsList = document.getElementById("measurementsList")
}

function showFieldInfo(fieldName) {
    drawingLayer.clearLayers();
    var field = fields[fieldName];
    console.log(fieldName);
    if (document.getElementsByClassName("fieldInfo").length >= 1) {
        document.getElementsByClassName("fieldInfo")[0].remove();
    }
    var fieldInfo = document.createElement('div')
    fieldInfo.classList.add("fieldInfo", "col-7")
```

```
    fieldInfo.innerHTML = "<span id='boxclose'></span>\
                           <h5 style='text-align:center;'>Информация по полю:</h5>\
                           <strong>Название: </strong>\
                           <span>" + field.name + "</span><br>\
                           <strong>Координаты: </strong><br>\
                           <span>" + field.coordinatesString + "</span><br>\
                           <strong>Создано: </strong>\
                           <span>" + field.createdAt + "</span><br>\
                           <strong>Последнее изменение: </strong>\
                           <span>" + field.changedAt + "</span>";
    document.getElementById("chooseField").appendChild(fieldInfo);
    boxclose.onclick = () => document.getElementsByClassName("fieldInfo")[0].remove();
    field.drawPolygon();
}


function isMarkerInsidePolygon(marker, poly) {
    var polyPoints = poly.getLatLngs()[0];
    var x = marker.getLatLng().lat,
        y = marker.getLatLng().lng;

    var inside = false;
    for (var i = 0, j = polyPoints.length - 1; i < polyPoints.length; j = i++) {
        var xi = polyPoints[i].lat,
            yi = polyPoints[i].lng;
        var xj = polyPoints[j].lat,
            yj = polyPoints[j].lng;
        var intersect = ((yi > y) != (yj > y)) &&
            (x < (xj - xi) * (y - yi) / (yj - yi) + xi);
        if (intersect) inside = !inside;

    }

    return inside;
};
```

ДОДАТОК З

Веб-інтерфейс візуалізації геопросторових даних для мікросервісної платформи

Опис програмного модулю

УКР.НТУУ”КПІ ім. Ігоря Сікорського”_ТЕФ_АПЕПС_ТВ51148_19Б 12-2

Аркушів 8

Київ – 2019

АНОТАЦІЯ

Додаток надає лаконічний та структурний опис реалізації модулю обробки геоінформаційних даних про поля та приведення їх до необхідного типу, що готовий до візуалізації на карті.

Методи сервісу надають можливість приймати дані з серверу та обробляти їх, визначати помилки в задані та приводити до правильного типу, що являє з себе масив координат довготи і широти.

Модуль було створено за допомогою мови програмування Javascript.

.

ЗМІСТ

-4-

ЗАГАЛЬНІ ВІДОМОСТІ

Програмний код модулю міститься в межах класу Field, що включає в себе методи для приведення типів, створення HTML коду з даними про поле для подальшого відображення на мапі.

Модуль є ядром програмного продукту, адже виконує операції над основною структурную одиницею системи — полем.

Необхідні дані про поля сервіс приймає з серверу на мові Java, який пов’язаний з базою даних PostgreSQL.

-5-

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Сервіс реалізує наступний функціонал:

— приведення строкових даних про координати до географічного типу з даними про широту і довготу;

— генерація HTML вікна з основними даними про поле;

— визначає потрапляння точкових вимірів в межі поля;

— визначення середнії значень щільності грунта в заданій області;

— відображення поля у вигляді полігону з вершинами на карті;

— забарвлення полігону в колір, що залежить від рівня щільності грунту.

-6-

ОПИС ЛОГІЧНОЇ СТРУКТУРИ

Сервіс реалізовано у формі класу, який моделює інформацю про реальне аграрне поле, на якому провели виміри щільності грунту, та має наступні методи для манупуляцій з ними:

1) checkMeasurements();
2) showMeasurements();
3) drawPolygon();
4) get fieldHtml;
5) stringToArrayOfPoints (String);
6) get avrgMeasurements();
7) get coordinatesString().

-7-

ТЕХНІЧНІ ЗАСОБИ, ЩО ВИКОРИСТОВУВАЛИСЯ

Програмний код модулю було написано у текствому редакторі Visual Studio Code з набором користувацьких плагінів та розширень.

Весь програмний код було написано на нативному JavaScript без використання фреймворків.

Для реалізації відображення на карті границь заданого поля використовувалась додаткова бібліотека Leaflet.

-8-

ВХІДНІ ТА ВИХІДНІ ДАНІ

Вхідними даними є:

— JSON-файл з даними про поле

Вихідними даними є:

— HTML-код для вставки у вспливаюче вікно на карті;

— середні значення щільності грунту на полі;

— масив географічних координат з даними про вершини полігону поля.